**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TESTNG\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**TestNG (Next Generation)**

in """selenium using java""" there are 2 types of testing framework available.

1) TestNG

2) JUnit

in """selenium using python"" there are 2 types of testing framework available.

1) py:unit

2) py:test

in """selenium using PHP""" there are 2 types of testing framework available.

1) Behat

2) Mink

in """selenium using C#""" there are 1 types of testing framework available.

1) Nunit

in ""selenium using Ruby""" there are 2 types of testing framework available.

1) Repect

2) Test:unit

TestNG (Next Generation)Testing Framework.

- TestNG is open source testing framework, anybody can download this tool free of cost.

- Initially TestNG is developed for unit Testing (White box testing) but now we use for functional and regression testing..

- TestNG is testing framework used from unit testing to system testing.

- TestNG is inspired tool from Junit and Nunit tool but introducing the new features in TestNG tool, and TestNG is more powerful tool in market.

- TestNG is easier to use as compared with other testing framework tools.

**Advantages of TestNG VVVVVIMMMPPPPP**

1) TestNG annotation are used to create test cases.

2) by using TestNG annotation we can group the test cases.

3) by using TestNG can prioritized the test cases.

4) by using TestNG we can run same test cases multiple times

5) by using TestNG we can run test cases within the time.

6) by using TestNG we can parametrization test cases

7) by using TestNG we can achieve the data driven testing.

8) TestNG automatically generate the HTML reports.

9) by using TestNG we can run test cases in parallel.

10) Easily we can integrate TestNG with maven build, Gradle build tool , ant build tool etc

**Note:**

1) main method is not used in TestNG

2) TestNG class contains we have only methods that contains @Test Annotation and PreCondition annotation and PostCondition annotation

3) If we don't mention the TestNG annotation then the methods will not get executed.

**TestNG contains we have 3 types of annotations**

1) PreCondition annotation

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

2) Test annotation

@Test

3) PostCondition annotation

@AfterMethod

@AfterClass

@AfterTest

@AfterSuite

**What is execution sequence of TestNG annotations? VVVVVVVVVVVVVIMMMPPPPPPPPPPPP**

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@AfterTest

@AfterSuite

@Test Annotation

- it is used to create the test cases in TestNG

- we define @Test Annotation above the non static method

- default priority for the @Test annotation is zero.

- if we mention multiple test cases by using @Test annotation then it will run test cases as per the alphabetical order.

- if we don't want to execute test cases as per the alphabetical order then we use priority attribute.

or

- if u want to maintain test execution flow then we use priority attribute.

syntax;

@Test

public void abc()

{

//code/statement

}

------

public class Demo1

{

@Test

public void xyz()

{

System.out.println("xyz TC");

}

@Test

public void abc()

{

System.out.println("abc TC");

}

@Test

public void pqr()

{

System.out.println("pqr TC");

}

}

package Tutorial1;

import org.testng.annotations.Test;

public class Sample1 {

@Test

public void pqr()

{

System.out.println("PQR Test Cases");

}

@Test

public void abc()

{

System.out.println("ABC Test Cases");

}

@Test

public void xyz()

{

System.out.println("XYZ Test Cases");

}

}

package Tutorial1;

import org.testng.annotations.Test;

public class Sample2 {

@Test(priority = 1)

public void registerPage()

{

System.out.println("Register Page Test Cases");//1

}

@Test(priority = 2)

public void loginPage()

{

System.out.println("Login Page Test Cases");//2

}

@Test(priority = 3)

public void homePage()

{

System.out.println("Home Page Test cases");//3

}

}

**What is priority attribute? VVVVVIMMMPPPP**

- it is used to maintain the test execution flow.

- default priority for test cases is zero

- we can mention negative priority for test cases

- we can mention same priority for multiple test cases.

**what is default priority for test cases? VVVVVVVIMMMPPPPPP**

- default priority for test cases is zero

**can we mention negative priority for test cases VVVVVVVIMMMPPPPPP**

yes we can

**can we mention same priority for multiple test cases VVVVVVVIMMMPPPPPP**

yes we can mention then it will execute the test cases as per the alphabetical order.

package Tutorial1;

import org.testng.annotations.Test;

public class Sample3 {

@Test(priority = -2)

public void abc() {

System.out.println("ABC Test Cases");

}

@Test(priority = 1)

public void lmn() {

System.out.println("LMN Test Cases");

}

@Test(priority = -1)

public void pqr() {

System.out.println("PQR Test Cases");

}

@Test(priority = 1)

public void xyz() {

System.out.println("XYZ Test Cases");

}

}

**How to run same test cases multiple times? VVVVVIMMMMPPPPPP**

- we can same test cases multiple times by using invocationCount attribute by passing count number

package Tutorial1;

import org.testng.annotations.Test;

public class Sample4 {

@Test(invocationCount = 10)

public void abc() {

System.out.println("ABC Test Cases");

}

@Test

public void pqr()

{

System.out.println("PQR Test Cases");

}

}

**How to run test cases within the time? VVVVVVVVVIMMMPPPPP**

- we use timeout attribute to run the test cases within the time by passing time in milliseconds.

package Tutorial1;

import org.testng.annotations.Test;

public class Sample4 {

@Test(timeOut = 5000)

public void abc() throws InterruptedException {

Thread.sleep(8000);

System.out.println("ABC Test Cases");

}

@Test

public void pqr() {

System.out.println("PQR Test Cases");

}

}

there are 2 ways we can maintain the test cases execution flow

1) by using priority attribute

2) by using dependsOnMethods attribute

2) by using dependsOnMethods attribute

i) hard dependsOnMethods attribute

- just imagine we have 3 test cases login test cases, homepage test case and logout test cases.

and we have mentioned logout test cases is dependsOnMethods as homepage test cases.

and if homepage test cases is fail then it will skip the logout test cases in dependsOnMethods attribute

ii) soft dependsOnMethods attribute

- to define the dependsOnMethods attribute then we use alwaysRun attribute with dependsOnMethods attribute.

- just imagine we have 3 test cases login test cases, homepage test case and logout test cases.

and we have mentioned logout test cases is soft dependsOnMethods as homepage test cases.

and if homepage test cases is fail then it will run the the logout test cases in soft dependsOnMethods attribute

syntax:

public class Sample5

{

@Test(dependsOnMethods={"pqr"})

public void abc()

{

System.out.println("ABC Test Cases");

}

@Test

public void pqr()

{

System.out.println("PQR Test Cases");

}

@Test(dependsOnMethods={"abc"})

public void xyz()

{

System.out.println("xyz Test Cases");

}

}

Execution sequence

PQR Test Cases

ABC test cases

XYZ test Cases

package Tutorial1;

import org.testng.annotations.Test;

public class Sample5 {

@Test(dependsOnMethods= {"pqr"})

public void abc() {

System.out.println("ABC Test Cases");

}

@Test

public void pqr() {

System.out.println("PQR Test Cases");

}

@Test(dependsOnMethods= {"abc"})

public void xyz() {

System.out.println("xyz Test Cases");

}

}

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

package Tutorial1;

import org.testng.annotations.Test;

public class Sample5 {

@Test

public void pqr() {

System.out.println("PQR Test Cases");//1

}

@Test(dependsOnMethods= {"pqr"})

public void home() {

System.out.println("Home Test Cases");//2

}

@Test(dependsOnMethods= {"pqr"})

public void login() {

System.out.println("login Test Cases");//3

}

}

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

package Tutorial1;

import org.testng.Assert;

import org.testng.annotations.Test;

public class Sample5 {

@Test

public void pqr() {

Assert.assertEquals(12, 3455);

System.out.println("PQR Test cases");

}

//Soft dependsOnMethods

@Test(dependsOnMethods= {"pqr"},alwaysRun=true)

public void home() {

System.out.println("Home Test Cases");//2

}

//hard dependsOnMethods

@Test(dependsOnMethods= {"home"})

public void login() {

System.out.println("login Test Cases");//3

}

}

public class Demo1

{

WebDriver driver;

@Test(priority=1)

public void openBrowserAndUrl()

{

//connect browser

//up casting

driver =new ChromeDriver();

//open a url

driver.get("https://www.saucedemo.com/");

}

@Test(priority=2)

public void loginFunctionality()

{

driver.findElement(By.id("user-name")).sendKeys("standard\_user");

driver.findElement(By.id("password")).sendKeys("secret\_sauce");

driver.findElement(By.id("login-button")).click();

}

@Test(priority=3)

public void addToCart()

{

driver.findElement(By.id("add-to-cart-sauce-labs-backpack")).click();

driver.findElement(By.id("add-to-cart-sauce-labs-bike-light")).click();

driver.findElement(By.xpath("//a[@class='shopping\_cart\_link']")).click();

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

}

@Test(priority=4)

public void enterPersonalInformation()

{

driver.findElement(By.id("first-name")).sendKeys("abc");

driver.findElement(By.id("last-name")).sendKeys("pqr");

driver.findElement(By.id("postal-code")).sendKeys("112211");

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

}

@Test(priority=5)

public void captureOrderDetails()

{

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

String a = driver.findElement(By.id("//h2[@class='complete-header']")).getText();

}

}

@Test

@Test(priority=1)

@Test(priority=-5)

@Test(invocationCount=10)

@Test(timeOut=500)

@Test(dependsOnMethods={"methodName"})

@Test(dependsOnMethods={"methodName"},alwaysRun=true)

What are the different annotations in TestNG? VVVVIMMMPPPPPPP

there are 3 types of annotations in TestNG.

1) Precondition annotation

i) @BeforeSuite

ii) @BeforeTest

iii) @BeforeClass

iv) @BeforeMethod

2) Test annotations

i) @Test

3) PostCondition annotation.

i) @AfterMethod

ii) @AfterClass

iii) @AfterTest

iv) @AfterSuite

1) @BeforeMethod annotations

-it is pre condition for each and every test cases in class.

syntax:

@BeforeMethod

public void abc()

{

//statement

}

2) @AfterMethod annotations

- it is post condition for each and every test cases in class.

syntax:

@AfterMethod

public void xyz()

{

//statement

}

Scenario 1:

@BeforeMethod

Test Cases 1

Test cases 2

Test cases 3

@AfterMethod

O/p

@BeforeMethod

test Cases 1

@AfterMethod

@BeforeMethod

Test cases 2

@AfterMethod

@BeforeMethod

Test cases 3

@AfterMethod

package Tutorial2;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class Sample1 {

@BeforeMethod

public void setUp() {

System.out.println("Before Method");

}

@Test

public void testCases1() {

System.out.println("Test Cases 1");

}

@Test

public void testCases2() {

System.out.println("Test Cases 2");

}

@Test

public void testCases3() {

System.out.println("Test cases 3");

}

@AfterMethod

public void tearDown() {

System.out.println("After Method");

System.out.println("-----------------------------");

}

}

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

**@BeforeClass**

- it is precondition for all the test cases in class

**@AfterClass**

- it is postcondition for all the test cases in class.

syntax:

@BeforeClass

public void a1()

{

//statement

}

@AfterClass

public void a2()

{

//statement

}

Scenario 1"

@BeforeClass

Test Cases1

Test Cases 2

test Cases 3

@AfterClass

o/p

@BeforeClass

Test Cases1

Test Cases 2

Test Cases 3

@AfterClass

package Tutorial2;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class Sample2 {

@BeforeClass

public void a1() {

System.out.println("BeforeClass");

}

@Test

public void testCases1() {

System.out.println("Test cases 1");

}

@Test

public void testCases2() {

System.out.println("Test cases 2");

}

@Test

public void testCases3() {

System.out.println("Test cases 3");

}

@AfterClass

public void b2() {

System.out.println("After Class");

}

}

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

**diff between @BeforeClass and @BeforeMethod VVVVVIMMPPPPP**

@BeforeClass is precondition for all the test cases in class.

@BeforeMethod is precondition for each and every test cases in class.

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

Scenario 3"

@BeforeClass

@BeforeMethod

TestCase1

TestCase2

TestCase3

@AfterMethod

@AfterClass

o/p

@BeforeClass

@BeforeMethod

Test Cases1

@AfterMethod

@BeforeMethod

Test Cases 2

@AfterMethod

@BeforeMethod

Test Cases 3

@AfterMethod

@AfterClass

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

package Tutorial2;

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 Sample3 {

@BeforeClass

public void a1() {

System.out.println("Before Class");

}

@BeforeMethod

public void b1() {

System.out.println("Before Method");

}

@Test

public void testCases1() {

System.out.println("Test cases 1");

}

@Test

public void testCases2() {

System.out.println("Test cases 2");

}

@Test

public void testCases3() {

System.out.println("Test cases 3");

}

@AfterMethod

public void b2() {

System.out.println("After Method");

System.out.println("-------------------");

}

@AfterClass

public void a2() {

System.out.println("After Class");

System.out.println("===================================");

}

}

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

**How to ignore test cases in TestNG? VVVVVIMMMPPPPPPPPPPPPP**

there are 3 different ways we can ignore the test cases in TestNG

1) by using @Ignore annotation

2) by using enabled=false attribute

3) by grouping the test cases and inside the XML file use exclude tag name and by passing group name.

**1) by using @Ignore annotation**

- use @Ignore annotation before test cases

@Ignore

@Test

public void testcase1()

{

//statement

}

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

**2) by using enabled=false attribute**

- we have to use enabled=false attribute in @Test annotations

@Test(enabled=false)

public void testCase1()

{

//statement

}

**---------------------------------------------------------------------------------------------------------**

**Note:**

**if we mention invocationCount as zero or negative number then it will ignoring the test cases**

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

package Tutorial2;

import org.testng.annotations.Ignore;

import org.testng.annotations.Test;

public class Sample4 {

@Test

public void testCases1() {

System.out.println("Test cases 1");

}

@Ignore

@Test

public void testCases2() {

System.out.println("Test cases 2");

}

@Test

public void testCases3() {

System.out.println("Test cases 3");

}

}

package Tutorial2;

import org.testng.annotations.Ignore;

import org.testng.annotations.Test;

public class Sample4 {

@Test

public void testCases1() {

System.out.println("Test cases 1");

}

@Test

public void testCases2() {

System.out.println("Test cases 2");

}

@Test(enabled = false)

public void testCases3() {

System.out.println("Test cases 3");

}

}

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

package Tutorial2;

import java.time.Duration;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class Sample5 {

WebDriver driver;

@BeforeClass

public void setUp() {

// connect browser

System.setProperty("webdriver.chrome.driver",

"C:\\Users\\praf0\\OneDrive\\Desktop\\Automation Testing 27th April 2024 Batch\\chromedriver.exe");

// up casting

driver = new ChromeDriver();

// implicit wait

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

// maximize

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

}

@Test

public void google() throws InterruptedException {

// open a google application

driver.get("https://www.google.com");

// capture title

System.out.println(driver.getTitle());

Thread.sleep(5000);

}

@Test

public void facebook() throws InterruptedException {

// open a facebook application

driver.get("https://www.facebook.com");

// capture title

System.out.println(driver.getTitle());

Thread.sleep(5000);

}

@AfterClass

public void tearDown() throws InterruptedException {

Thread.sleep(5000);

// quit browser

driver.quit();

}

}

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

**@BeforeTest**  **VVVVIMMMPPPPPPP**

**- it is precondition for all the test cases in classes tags in xml file**

**@AfterTest**

**- it is post condition for all the test cases in classes tags in xml file**

syntax

@BeforeTest

public void a1()

{

//statement

}

@AfterTest

public void a2()

{

//statement

}

what are the different tags in xml file VVVVIMMMPPPPPPP

<suite name="suite name or project name">

<test name="regression test cases">

<classes>

<class name="packagename.classname1"> </class>

<class name="packagename.classname2"> </class>

</classes>

</test>

</suite>

<suite name="suite name or project name">

<test name ="test cases name"

<classes>

<class name ="packagename.classname"> </class>

</classes>

</test>

</suite>

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

**Execute multiple classes using XML (Extensive Markup Language) File**

Scenario

Tutorial2

create sample7 class

-- create the 2 test cases

create sample8 class

-- create 3 test cases

create sample9 class

-- create 4 test cases

- run three classes

<suite name="Orange HRM">

<test name="test cases">

<classes>

<class name="Tutorial2.Sample7"> </class>

<class name="Tutorial2.Sample9"> </class>

<class name="Tutorial2.Sample8"> </class>

</classes>

</test>

<suite>

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

Sample7 class

@BeforeTest

s1 test cases

s2 test cases

@AfterTest

Pune7 class

P1 test cases

p2 test cases

p3 test cases

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

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

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

<suite name="Suite">

<test name="Test">

<classes>

<class name="Tutorial2.Pune7"> </class>

<class name="Tutorial2.Sample7"></class>

</classes>

</test> <!--

Test -->

</suite> <!--Suite -->

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

o/p

@BeforeTest

p1

P2

P3

S1

S2

@AfterTest

package Tutorial2;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class Sample7 {

@BeforeTest

public void a1() {

System.out.println("Before Test");

}

@Test

public void s1() {

System.out.println("S1 Test Cases");

}

@Test

public void s2() {

System.out.println("S2 Test Cases");

}

@AfterTest

public void a2() {

System.out.println("After Test");

}

}

package Tutorial2;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class Pune7 {

@Test

public void p1() {

System.out.println("p1 Test Cases");

}

@Test

public void p2() {

System.out.println("p2 Test Cases");

}

@Test

public void p3() {

System.out.println("p3 Test Cases");

}

}

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

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

<suite name="Suite">

<test name="Test">

<classes>

<class name="Tutorial2.Pune7"> </class>

<class name="Tutorial2.Sample7"></class>

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

======

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

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

<suite name="Suite">

<test name="Test">

<classes>

<class name="Tutorial2.Pune7"> </class>

<class name="Tutorial2.Sample7"></class>

<class name="Tutorial2.Mumbai7"></class>

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

package Tutorial2;

import org.testng.annotations.Test;

public class Mumbai7 {

@Test

public void m1() {

System.out.println("M1 Test Cases");

}

@Test

public void m2() {

System.out.println("M2 Test Cases");

}

@Test

public void m3() {

System.out.println("M3 Test Cases");

}

}

=

package Tutorial8;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class Sample8 {

@BeforeMethod

public void a1() {

System.out.println("Before Method");

}

@Test

public void s1() {

System.out.println("S1 Test cases");

}

@Test

public void s2() {

System.out.println("S2 Test cases");

}

@AfterMethod

public void a2() {

System.out.println("After Method");

}

}

package Tutorial8;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class Pune8 {

@BeforeClass

public void b1() {

System.out.println("Before Class");

}

@Test

public void p1() {

System.out.println("P1 Test cases");

}

@Test

public void p2() {

System.out.println("P2 Test cases");

}

@AfterClass

public void b2() {

System.out.println("After class");

}

}

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

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

<suite name="Suite">

<test thread-count="5" name="Test">

<classes>

<class name="Tutorial8.Pune8" />

<class name="Tutorial8.Sample8" />

</classes>

</test> <!--

Test -->

</suite>

<!--

@BeforeClass

P1 test cases

P2 Tets cases

@After Class

@BeforeMethod

s1 Test cases

@AfterMethod

@BeforeMethod

S2 test cases

@AfterMethod

-->

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

package Tutorial8;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class Mumbai8 {

@BeforeTest

public void c1() {

System.out.println("Before test");

}

@Test

public void m1() {

System.out.println("M1 Test Cases");

}

@Test

public void m2() {

System.out.println("M2 Test Cases");

}

@AfterTest

public void c2() {

System.out.println("After test ");

}

}

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

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

<suite name="Suite">

<test thread-count="5" name="Test">

<classes>

<class name="Tutorial8.Pune8" />

<class name="Tutorial8.Sample8" />

<class name="Tutorial8.Mumbai8" />

</classes>

</test> <!--

Test -->

</suite>

<!--

@BeforeTest

@BeforeClass

P1

P2

AfterClass

@BeforeMethod

S1

@AfterMethod

@BeforeMethod

s2

AfterMethod

m1

m2

@AfterTest

-->

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

**diff between @BeforeClass and @BeforeTest VVVVVIMMMPPPPPP**

@BeforeClass is precondition for all the test cases in class

@BeforeTest is pre condition for all the test cases in classes tags in xml file

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

package Tutorial8;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class Sample2 {

@BeforeMethod

public void abc() {

System.out.println("Before Method ABC ::: 1");

}

@BeforeMethod

public void aaa() {

System.out.println("Before Method AAA ::: 1");

}

@Test

public void s1() {

System.out.println("S1 Test cases ::: 3");

}

@AfterMethod

public void bbb() {

System.out.println("After Method BBB ::: 4");

}

@AfterMethod

public void ccc() {

System.out.println("After Method CCC ::: 5");

}

}

package Tutorial8;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class Sample2 {

@BeforeMethod

public void aaa() {

System.out.println("Before Method AAA ::: 1 & 6");

}

@BeforeMethod

public void abc() {

System.out.println("Before Method ABC ::: 2 & 7");

}

@Test

public void s1() {

System.out.println("S1 Test cases ::: 3");

}

@Test

public void s2() {

System.out.println("S2 Test cases ::: 8");

}

@AfterMethod

public void bbb() {

System.out.println("After Method BBB ::: 4 & 9");

}

@AfterMethod

public void ccc() {

System.out.println("After Method CCC ::: 5 & 10");

}

}

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

package Tutorial8;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class Sample2 {

@BeforeMethod

public void aaa() {

System.out.println("Before Method AAA ::: 1");

}

@BeforeMethod

public void abc() {

System.out.println("Before Method ABC ::: 2");

}

@Test

public void s1() {

System.out.println("S1 Test cases ::: 3");

}

}

====

Orange HRM application

@BeforeClass

- open a browser and url

test Cases 1

- login functionality

test cases 2

- click on pim link

click on add employee button

enter firstname, last name, and click on save button

test cases 3

- capture the employee id -------------> store employee id globally

- Enter SSN Number

- Select DOJ

- select country

- select gender

- click on save button

test cases 4:

- click on employee list

- enter captured employee id

- click on search button

test cases 5:

click on profile image

click on logout

@AfterMethod

- takes screenshot code with date and time

@AfterClass

quit the browser

=

package Tutorial8;

import org.testng.annotations.AfterClass;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class OrangeHRMTest {

@BeforeClass

public void setUp() {

// Connect to actual browser

// up casting ---> declare webdriver globally

// implicit wait

// maximize

// open a url

}

@Test(priority = 1)

public void loginFunctionality() {

// find the username text and enter value

// find the password text and enter value

// find and click on login button

}

@Test(priority = 2)

public void addNewEmployeeInPIMPage() {

// find and click on PIM Link

// find and click on add employee button

// find and enter first name and lastname

// find and click on save button

}

@Test(priority = 3)

public void addMoreDetails() {

// find and capture the employee id -------------> store employee id globally

// find and select country

// find and select gender

// find and click on save button

}

@Test(priority = 4)

public void searchEmployee() {

// find and click on employee list

// find and enter captured employee id

// find and click on search button

}

@Test(priority = 5)

public void logoutFunctionality() {

// find and click on profile image

// find and click on logout

}

@AfterMethod

public void tearDown() {

// convert WebDriver Object into TakesScreenshot

// capture actual screenshot

// create object of SimpleDateFormat class by passing date and time format

// capture the current date and time in format

// store screenshot in destination location

// copy screenshot from source to destination location

}

@AfterClass

public void abc() {

// close the session

}

}

**What is use of @Parameters Annotation in TestNG? VVVIIMMMPPPPPPPPPP**

**or**

**How to achieve parameterization in TestNG? VVVIIMMMPPPPPPPPPP**

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

- we achieve parameterization in TestNG using """"@Parameters annotations.""""

- we can use @Parameters annotation in Test Cases Level as Precondition level.

- @Parameters annotation it fetch the values/input/test data from XML file, and we can use these values

inside the test cases or inside the preconditions.

- we use @Parameters annotation to achieve the parallel testing as well as cross browser testing.

- to achieve the cross browser testing or parallel testing we pass browser names/values from the XML file.

syntax:

@Parameters({"parametername"})

@Test

public void abc(String variableName)

{

//statement

}

e.g.

<suite name="suite name or project name">

<parameter name="parametername" value="parametervalue" />

<test name="test cases name">

<classes>

<class name="packagename.classname"/>

</classes>

</test>

</suite>

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

e.g. Pass username from xml file

<suite name="orange HRM">

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

<parameter name="password" value="admin123" />

<test name="Login Test Cases">

<classes>

<class name="Tutorial3.Sample1"/>

</classes>

</test>

</suite>

@Parameters({"username","password"})

@Test

public void loginFunctionality(String abc, String xyz)

{

System.out.println(abc);

System.out.println(xyz);

}

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

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

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

<suite name="Suite">

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

<test name="Test">

<classes>

<class name="Tutorial3.Sample1" />

</classes>

</test>

</suite>

package Tutorial3;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

public class Sample1 {

@Parameters({"username"})

@Test

public void loginFunctionality(String abc)

{

System.out.println(abc);

}

}

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

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

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

<suite name="Suite">

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

<parameter name="password" value="admin123" />

<test name="Test">

<classes>

<class name="Tutorial3.Sample1" />

</classes>

</test>

</suite>

package Tutorial3;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

public class Sample1 {

@Parameters({ "username", "password" })

@Test

public void loginFunctionality(String abc, String xyz) {

System.out.println(abc);

System.out.println(xyz);

}

}

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

Scenario:

open a orange sauce demo application

- pass username and password from XML File

<suite name="Sauce Demo Application">

<parameter name="uname" value="standard\_user" />

<parameter name="pass" value="secret\_sauce"> </parameter>

<test name="login Page Test Cases">

<classes>

<class name="Tutorial3.Sample3" />

</classes>

</test>

</suite>

public class Sample3

{

@Parameters({"uname", "pass"})

@Test

public void loginFunctionality(String uname, String pass)

{

//Connect to actual browser

//Up Casting

WebDriver driver =new ChromeDriver();

//open a url

driver.get("https://www.saucedemo.com/");

//find and enter the value in username text box

WebElement unamewb = driver.findElement(By.id("user-name"));

unamewb.sendKeys(uname);

//find and enter value in password text box

WebElement passwb = driver.findElement(By.id("password"));

passwb.sendKeys(pass);

//find and click on login button

WebElement loginwb = driver.findElement(By.id("login-button"));

loginwb.click();

}

}

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

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

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

<suite name="Suite">

<parameter name="uname" value="performance\_glitch\_user"></parameter>

<parameter name="pass" value="secret\_sauce"></parameter>

<test name="Test">

<classes>

<class name="Tutorial3.Sample3" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

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

Scenario: https://ui.cogmento.com/

Open a application in chrome browser

or

Open a application in edge browser

or

Open a application in incognito browser

<suite name="Cogmento CRM Application">

<parameter name="browsername" value="incognito" />

<test name="Browser Functionality">

<classes>

<class name="Tutorial3.Sample4"/>

</classes>

</test>

</suite>

public class Sample4{

WebDriver driver;

@Parameters({"browsername"})

@Test

public void openBrowser(String browsername)

{

if(browsername.equals("chrome"))

{

//connect to chrome browser

//up casting

driver =new ChromeDriver();

}

else if(browsername.equals("edge"))

{

//connect to edge browser

//up casting

driver=new EdgeDriver();

}

else if(browsername.equals("incognito"))

{

//connect to actual chrome browser

//create object of ChromeOptions class

ChromeOptions opt =new ChromeOptions();

// use addArguments() method by passing --incognito keyword

opt.addArguments("--incognito");

//up casting by passing ChromeOptions object name.

driver =new ChromeDriver(opt);

}

else

{

System.out.println("Please Enter valid browser name");

}

driver.get("https://ui.cogmento.com/");

}

}

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

package Tutorial3;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.edge.EdgeDriver;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

public class Sample4 {

WebDriver driver;

@Parameters({ "browsername" })

@Test

public void openBrowser(String browsername) {

if (browsername.equals("chrome")) {

// connect to chrome browser

// up casting

driver = new ChromeDriver();

} else if (browsername.equals("edge")) {

// connect to edge browser

// System.setProperty("webdriver.edge.driver", "path of edge driver");

// up casting

driver = new EdgeDriver();

} else if (browsername.equals("incognito")) {

// connect to actual browser

// create object of ChromeOptions class

ChromeOptions opt = new ChromeOptions();

// use addArguments()method by passing --incognito keyword

opt.addArguments("--incognito");

// up casting by passing ChromeOptions object name

driver = new ChromeDriver(opt);

} else {

System.out.println("Please Enter valid Browser name");

}

driver.get("https://ui.cogmento.com/");

}

}

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

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

<suite name="Cogmento CRM">

<parameter name="browsername" value="edge"></parameter>

<test name="Browser Test Cases">

<classes>

<class name="Tutorial3.Sample4" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

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

public class PIMPageTest

{

WebDriver driver;

String employeeId;

@Parameters({"browsername"})

@BeforeClass

public void setup(String browsername)

{

if(browsername.equals("chrome"))

{

//connect to chrome browser

//up casting

driver =new ChromeDriver();

}

else if(browsername.equals("edge"))

{

//connect to edge browser

//up casting

driver = new EdgeDriver();

}

else if(browsername.equals("incognito"))

{

//connect to actual browser

//create object of ChromeOptions class

ChromeOptions opt =new ChromeOptions();

//use addArguments() method by passing --incognito keyword

opt.addArguments("--incognito");

//up casting concept by passing ChromeOptions object name

driver =new ChromeDriver(opt);

}

else

{

System.out.println("Please enter valid browser name");

}

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

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

}

@Test(priority=1)

public void loginFunctionality()

{

Thread.sleep(7000);

driver.findElement(By.name("username")).sendKeys("Admin");

driver.findElement(By.name("password")).sendKeys("admin123");

driver.findElement(By.xpath("//button[text()=' Login ']")).click();

}

@Test(priority=2)

public void addNewEmployeeInPIMPage()

{

driver.findElement(By.xpath("//span[text()='PIM']")).click();

driver.findElement(By.xpath("//a[text()='Add Employee']")).click();

driver.findElement(By.name("firstName")).sendKeys("Admin");

driver.findElement(By.name("lastName")).sendKeys("Admin");

driver.findElement(By.xpath("//button[text()=' Save ']")).click();

}

@Test(priority=3)

public void addMoreDetails()

{

employeeId = driver.findElement(By.xpath("//label[text()='Employee Id']/parent::div/following-sibling::div/child::input")).getText();

driver.findElement(By.xpath("//label[text()='Nationality']/parent::div/following-sibling::div/descendant::div[@class='oxd-select-text-input']")).click();

List<WebElement> ls = driver.findElements(By.xpath("//div[@class='oxd-select-option']/child::span"));

for(WebElement abc: ls)

{

String country = abc.getText();

if(country.equals("Indian"))

{

abc.click();

break;

}

}

driver.findElement(By.xpath("//label[text()='Gender']/parent::div/following-sibling::div/descendant::input[@value='1']")).click();

driver.findElement(By.xpath("//p[text()=' \* Required']/following-sibling::button")).click();

}

@Test(priority=4)

public void searchEmployee()

{

driver.findElement(By.xpath("//a[text()='Employee List']")).click();

driver.findElement(By.xpath("//label[text()='Employee Id']/parent::div/following-sibling::div/child::input")).sendKeys(employeeId);

driver.findElement(By.xpath("//button[text()=' Search ']")).click();

}

@Test(priority=5)

public void logOutFunctionality()

{

driver.findElement(By.xpath("//p[@class='oxd-userdropdown-name']")).click();

driver.findElement(By.xpath("//a[text()='Logout']")).click();

}

@AfterMethod

public void abc()

{

TakesScreenshot ts = (TakesScreenshot)driver;

File src = ts.getScreenshotAs(OutputType.FILE);

String date = new SimpleDateFormat("ddMMyyyy\_HHmmss").format(new Date());

File dest =new File(System.getProperty("user.dir")+"\\MayPassScreenshot\\orangehrm"+date+".png");

FileUtils.copyFile(src,dest);

}

@AfterClass

public void teardown()

{

driver.quit();

}

}

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

package Tutorial4;

import java.io.File;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.time.Duration;

import java.util.Date;

import java.util.List;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.edge.EdgeDriver;

import org.testng.annotations.AfterClass;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

public class PIMPageTest {

WebDriver driver;

String employeeId;

@Parameters({ "browsername" })

@BeforeClass

public void setup(String browsername) {

if (browsername.equals("chrome")) {

// connect to chrome browser

// up casting

driver = new ChromeDriver();

} else if (browsername.equals("edge")) {

// connect to edge browser

// up casting

driver = new EdgeDriver();

} else if (browsername.equals("incognito")) {

// connect to actual browser

// create object of ChromeOptions class

ChromeOptions opt = new ChromeOptions();

// use addArguments() method by passing --incognito keyword

opt.addArguments("--incognito");

// up casting concept by passing ChromeOptions object name

driver = new ChromeDriver(opt);

} else {

System.out.println("Please enter valid browser name");

}

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

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

}

@Test(priority = 1)

public void loginFunctionality() throws InterruptedException {

Thread.sleep(7000);

driver.findElement(By.name("username")).sendKeys("Admin");

driver.findElement(By.name("password")).sendKeys("admin123");

driver.findElement(By.xpath("//button[text()=' Login ']")).click();

}

@Test(priority = 2)

public void addNewEmployeeInPIMPage() {

// Thread.sleep(7000);

driver.findElement(By.xpath("//span[text()='PIM']")).click();

driver.findElement(By.xpath("//a[text()='Add Employee']")).click();

driver.findElement(By.name("firstName")).sendKeys("Ashwini");

driver.findElement(By.name("lastName")).sendKeys("Patil");

employeeId = driver

.findElement(By.xpath("//label[text()='Employee Id']/parent::div/following-sibling::div/child::input"))

.getAttribute("value");

System.out.println(employeeId);

driver.findElement(By.xpath("//button[text()=' Save ']")).click();

}

@Test(priority = 3)

public void addMoreDetails() throws InterruptedException {

Thread.sleep(3000);

driver.findElement(By.xpath(

"//label[text()='Nationality']/parent::div/following-sibling::div/descendant::div[@class='oxd-select-text-input']"))

.click();

List<WebElement> ls = driver.findElements(By.xpath("//div[@class='oxd-select-option']/child::span"));

for (WebElement abc : ls) {

String country = abc.getText();

Thread.sleep(10);

if (country.equals("Indian")) {

abc.click();

break;

}

}

Thread.sleep(4000);

WebElement radioButton = driver.findElement(By

.xpath("//label[text()='Gender']/parent::div/following-sibling::div/descendant::label[text()='Male']"));

((JavascriptExecutor) driver).executeScript("arguments[0].click();", radioButton);

Thread.sleep(4000);

driver.findElement(By.xpath("//p[text()=' \* Required']/following-sibling::button")).click();

}

@Test(priority = 4)

public void searchEmployee() throws InterruptedException {

driver.findElement(By.xpath("//a[text()='Employee List']")).click();

driver.findElement(By.xpath("//label[text()='Employee Id']/parent::div/following-sibling::div/child::input"))

.sendKeys(employeeId);

Thread.sleep(4000);

driver.findElement(By.xpath("//button[text()=' Search ']")).click();

}

@Test(priority = 5)

public void logOutFunctionality() throws InterruptedException {

driver.findElement(By.xpath("//p[@class='oxd-userdropdown-name']")).click();

Thread.sleep(4000);

driver.findElement(By.xpath("//a[text()='Logout']")).click();

}

@AfterMethod

public void abc() throws IOException, InterruptedException {

Thread.sleep(4000);

TakesScreenshot ts = (TakesScreenshot) driver;

File src = ts.getScreenshotAs(OutputType.FILE);

String date = new SimpleDateFormat("ddMMyyyy\_HHmmss").format(new Date());

File dest = new File(System.getProperty("user.dir") + "\\MayPassScreenshot\\orangehrm" + date + ".png");

FileUtils.copyFile(src, dest);

}

@AfterClass

public void teardown() {

driver.quit();

}

}

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

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

<suite name="Suite">

<parameter name="browsername" value="incognito" />

<test name="Test">

<classes>

<class name="Tutorial4.PIMPageTest" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

**What are the different assertion in TestNG? VVVVVIMMMPPPPPPPPP**

**or**

**What is difference between Assert and Verify? VVVVVIMMMPPPPPPPPPPPPPPPPPPPPPP**

**or**

**What is difference between Hard Assert and Soft Assert? VVVVVIMMMPPPPPPPPP**

**there are 2 types of assertion in TestNG**

**1) Hard Assert**

**2) Soft Assert.**

**1) Hard Assert**

- Hard Assert is also called as Assert.

- Hard assert is type of assertion which throws exception immediately when assert statement are fail and it will execute next test cases.

- Hard assert is type of assertion which throws exception immediately when assert statement are fail and it will not going to execute remaining of line of code

after assertion failure.

- All Hard assert methods are static method,

- In Hard Assert we compare the byte, short, int, long, double ,float, boolean, char, String and Arrays.

syntax:

Assert.assertEquals("ActualValue", "ExpectedValues");

**2) Soft Assert**.

- Soft Assert is also called as Verify.

- Soft Assert it collect the Exceptions during the executions,

- in Soft Assert if any assertion is fail, it will execute the remaining line of code after the assertion failure.

- Soft Assert it does not throws exception immediately, it collects the error/ exception during the execution.

- If u want to use Soft Assertion then we have to create object of SoftAssert class.

- and then by using SoftAssert object name we can assertEquals()method to compare the byte, int, short, long, float, double, char, boolean, String , arrays.

- if u want to throws the exception after assertion failure then we use assertAll() methods. VVVVVIMMMMPPPPPP

- after using assertAll() method, if any exception occurs, it throws exception immediately.

syntax:

SoftAssert soft=new SoftAssert();

soft.assertEquals("actualValue", "expectedValue");

public class Demo2

{

@Test

public void a1()

{

System.out.println("A1 Test cases is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals(12,12);

System.out.println("A1 Test cases is Ended");

}

@Test

public void a2()

{

System.out.println("A2 Test cases is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals("facebook", "amazon"); //Soft Assert does not throws Exception immediately it collects the exceptions during the execution.

//as well as it run remaining lines of code after assertion failure.

System.out.println("A2 Test cases is Ended");

}

@Test

public void a3()

{

System.out.println("A3 Test cases is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals(12.34f, 12.34f);

System.out.println("A3 Test cases is Ended");

}

}

public class Demo1

{

@Test

public void a1()

{

System.out.println("A1 Test cases is started");

Assert.assertEquals(12,12);

System.out.println("A1 Test cases is Ended");

}

@Test

public void a2()

{

System.out.println("A2 Test cases is started");

Assert.assertEquals("SauceDemo","OrangeHRM); //immediately it will throws exception and it goes to next test cases.

System.out.println("A2 Test cases is Ended");

}

@Test

public void a3()

{

System.out.println("A3 Test cases is started");

Assert.assertEquals(12.3f,12.3f);

System.out.println("A3 Test cases is Ended");

}

}

package Tutorial5;

import org.testng.annotations.Test;

import org.testng.asserts.SoftAssert;

public class Demo102 {

@Test

public void a1() {

System.out.println("A1 TC is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals(12, 12);

System.out.println("A1 TC is ended");

}

@Test

public void a2() {

System.out.println("A2 TC is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals("facebook", "google");

soft.assertAll();

System.out.println("A2 TC is ended");

//soft.assertAll();

}

@Test

public void a3() {

System.out.println("A3 TC is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals(122.23f, 122.23f);

System.out.println("A3 TC is ended");

}

}

package Tutorial5;

import org.testng.Assert;

import org.testng.annotations.Test;

import org.testng.asserts.SoftAssert;

public class Demo103 {

@Test

public void d1() {

System.out.println("D1 TC is started");//1

Assert.assertEquals(12, 12);//Hard assertion

System.out.println("D1 TC is Ended");//2

}

@Test

public void d2() {

System.out.println("D2 TC is started");//3

Assert.assertEquals("facebook", "google");//Hard Assert

System.out.println("D2 TC is Ended");

}

@Test

public void d3() {

System.out.println("D3 TC is started");//4

SoftAssert soft = new SoftAssert();

soft.assertEquals("gmail", "outlook");//Soft Assert

System.out.println("D3 TC is Ended");//5

}

@Test

public void d4() {

System.out.println("D4 TC is started"); //6

SoftAssert soft = new SoftAssert();

soft.assertEquals(12, 9099900);//Soft Assert

System.out.println("D4 TC is Running");//7

Assert.assertEquals(true, false);//Hard Assert

System.out.println("D4 TC is Ended");

}

@Test

public void d5() {

System.out.println("D5 TC is started");//8

SoftAssert soft = new SoftAssert();

soft.assertEquals(12, 9099900); //Soft Assert

System.out.println("D5 TC is Running");//9

soft.assertAll();

System.out.println("D5 TC is Ended");

}

}

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

package Tutorial5;

import org.testng.Assert;

import org.testng.annotations.Test;

import org.testng.asserts.SoftAssert;

public class Demo104 {

@Test

public void a1() {

System.out.println("A1 TC is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals("gmail", "outlook");

System.out.println("A1 Tc is running");

System.out.println("A1 TC is Ended");

}

@Test

public void a2() {

System.out.println("A2 TC is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals("facebook", "google");

soft.assertAll();

System.out.println("A2 Tc is running");

System.out.println("A2 TC is Ended");

}

@Test

public void a3() {

System.out.println("A3 TC is started");

SoftAssert soft = new SoftAssert();

soft.assertEquals("selenium", "cucumber");

System.out.println("A3 Tc is running");

System.out.println("A3 TC is Ended");

soft.assertAll();

}

@Test

public void a4() {

System.out.println("A4 TC is started");

Assert.assertEquals("maven", "pom");

System.out.println("A4 Tc is running");

System.out.println("A4 TC is Ended");

}

@Test

public void a5() {

System.out.println("A5 TC is started");

SoftAssert soft = new SoftAssert();

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

System.out.println("A5 Tc is running");

System.out.println("A5 TC is Ended");

}

}

package Tutorial5;

import org.testng.Assert;

import org.testng.annotations.Test;

public class Demo101 {

@Test

public void a1() {

System.out.println("A1 Test cases is started");

Assert.assertEquals(12, 12);

System.out.println("A1 Test cases is ended");

}

@Test

public void a2() {

System.out.println("A2 Test cases is started");

Assert.assertEquals("google", "facebook");

System.out.println("A2 test cases is ended");

System.out.println("A2 test cases is ended");

System.out.println("A2 test cases is ended");

System.out.println("A2 test cases is ended");

}

@Test

public void a3() {

System.out.println("A3 Test cases is started");

Assert.assertEquals(true, true);

System.out.println("A3 test cases is ended");

}

}

===

If we have 100 test cases, and I want to run the only 80 Test cases.

or

If we have 100 test cases, and I want to skip the 20 Test cases.

or

how to run the specific groups

or

how to execute the groups

or

how to skip specific group.

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

how to create group for test cases?

- we use groups attribute by using group name in test level

syntax:

@Test(groups={"SanityTesting"})

public void d1()

{

}

can we mention 2 group name for same test cases?

Yes

syntax:

@Test(groups={"SanityTesting" , "RegressionTesting"})

public void d2()

{

}

**How to the specific group test cases?**

there are 2 ways we can run the groups test cases

1) by using Run Configuration

2) by using XML File

**1) by using Run Configuration**

- if u want to run the group test cases using Run Configuration then we right click on class, go to Run Configuration Option then click in group name radio button and select the group name checkbox and click apply and run..

**2) by using XML File**

- we use group tagname then run tagname and then include and exclude tags name after the test tags in xml file.

<suite name="Suite name /project name">

<test name="test cases name">

<group>

<run>

<include name="group name" />

</run>

</group>

<classes>

<class name="packagename.classname"/>

</classes>

</test>

</suite>

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

<suite name="suite name or project name ">

<test name="test case name ">

<groups>

<run>

<include name="group name " />

</run>

</group>

<classes>

<class name="PackageName.className" />

</classes>

</test>

</suite>

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

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

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

<suite name="Suite">

<test name="Test">

<groups>

<run>

<include name="SanityTesting"></include>

</run>

</groups>

<classes>

<class name="Tutorial5.Demo105" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

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

package Tutorial5;

import org.testng.annotations.Test;

public class Demo105 {

@Test(groups = { "SanityTesting" })

public void a1()

{

System.out.println("A1 TC Sanity Testing Group");

}

@Test(groups= {"SanityTesting","RegressionTesting"})

public void a2() {

System.out.println("A2 TC ::: Sanity and Regression Testing Group");

}

@Test(groups= {"PIMPage"})

public void a3()

{

System.out.println("A3 TC :: PIMPage group");

}

@Test(groups = {"Pranjali"})

public void a4()

{

System.out.println("A4 TC :: Pranjali Group");

}

}

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

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

<suite name="Suite">

<test name="Test">

<groups>

<run>

<exclude name="PIMPage"></exclude>

</run>

</groups>

<classes>

<class name="Tutorial5.Demo105" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

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

**If we have 100 test cases, and I want to run the only 80 Test cases. VVVVIMMPPPPPPPPPP**

**or**

**If we have 100 test cases, and I want to skip/Ignore the 20 Test cases. VVVVIMMPPPPPPPPPP**

**or**

**how to run the specific groups VVVVIMMPPPPPPPPPP**

**or**

**how to execute the groups**

**or**

**how to skip specific group.**

**how to create group for test cases?**

- we use groups attribute by using group name in test level

syntax:

@Test(groups={"SanityTesting"})

public void d1()

{

}

can we mention 2 group name for same test cases?

Yes

syntax:

@Test(groups={"SanityTesting" , "RegressionTesting"})

public void d2()

{

}

How to the specific group test cases?

there are 2 ways we can run the groups test cases

1) by using Run Configuration

2) by using XML File

**1) by using Run Configuration**

- if u want to run the group test cases using Run Configuration then we right click on class, go to Run Configuration Option then click in group name radio button and select the group name checkbox and click apply and run..

**2) by using XML File**

- we use group tagname then run tagname and then include and exclude tags name after the test tags in xml file.

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

Scenario:

Create Demo1 class

- create d1 test case and create group as SanityTesting

- create d2 test case and create group as RegressionTesting

- create d3 test case and create group as SystemTesting

create Sample1 class

- create s1 test cases and create group as SanityTesting

- create s2 test cases and create group as RegressionTesting

- create s3 test case and create group as SystemTesting

create Pune1 class

- create p1 test cases and create group as SanityTesting

- create p2 test cases and create group as RegressionTesting

- create p3 test case and create group as SystemTesting

1. Run all the test cases from SanityTesting group

XML(Extensive Markup Language) File

<suite name="Project Name">

<test name="Test case Name">

<groups>

<run>

<include name="SanityTesting"/>

</run>

</groups>

<classes>

<class name="Tutorial1.Demo1" />

<class name="Tutorial1.Sample1"/>

<class name="Tutorial1.Pune1"/>

</classes>

</test>

</suite>

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

2. Run all the test cases from SanityTesting group as well as RegressionTesting group

<suite name="Project Name">

<test name="Test case Name">

<groups>

<run>

<include name="SanityTesting"/>

<include name="RegressionTesting"/>

</run>

</groups>

<classes>

<class name="Tutorial1.Demo1" />

<class name="Tutorial1.Sample1"/>

<class name="Tutorial1.Pune1"/>

</classes>

</test>

</suite>

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

3. don't execute test cases from SystemTesting group

<suite name="project name">

<test name="test case name">

<groups>

<run>

<exclude name="SystemTesting" />

</run>

</groups>

<classes>

<class name="Tutorial1.Demo1" />

<class name="Tutorial1.Sample1"/>

<class name="Tutorial1.Pune1"/>

</classes>

</test>

</suite>

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

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

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

<suite name="Suite">

<test name="Test">

<groups>

<run>

<include name="SanityTesting"></include>

</run>

</groups>

<classes>

<class name="Tutorial1.Pune1" />

<class name="Tutorial1.Demo1" />

<class name="Tutorial1.Sample1" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

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

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

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

<suite name="Suite">

<test name="Test">

<groups>

<run>

<exclude name="SystemTesting"></exclude>

</run>

</groups>

<classes>

<class name="Tutorial1.Sample1" />

<class name="Tutorial1.Demo1" />

<class name="Tutorial1.Pune1" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

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

@Test(groups={"loginPage"}

public void abc1()

{

//statement

}

@Test(groups={"homePage"} , dependsOnGroups={"loginPage"})

public void abc2()

{

//statement

}

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

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

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

<suite name="Suite">

<test name="Test">

<groups>

<run>

<include name="RegressionTesting"></include>

<include name="SanityTesting"></include>

</run>

</groups>

<classes>

<class name="Tutorial1.Sample1" />

<class name="Tutorial1.Demo1" />

<class name="Tutorial1.Pune1" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

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

package Tutorial1;

import org.testng.annotations.Test;

public class Demo1 {

@Test(groups = { "SanityTesting","End2EndTesting" })

public void d1() {

System.out.println("D1 Test cases ::: SanityTesting group");

}

@Test(groups = { "RegressionTesting","End2EndTesting" }, dependsOnGroups = { "SanityTesting" })

public void d2() {

System.out.println("D2 Test Cases ::: RegressionTesting Group");

}

@Test(groups = { "SystemTesting","End2EndTesting" })

public void d3() {

System.out.println("D3 Test Cases ::: SystemTesting Group");

}

}

package Tutorial1;

import org.testng.annotations.Test;

public class Sample1 {

@Test(groups = { "SanityTesting","End2EndTesting" })

public void s1() {

System.out.println("S1 Test cases ::: SanityTesting group");

}

@Test(groups = { "RegressionTesting","End2EndTesting" }, dependsOnGroups= {"SanityTesting"})

public void s2() {

System.out.println("S2 Test Cases ::: RegressionTesting Group");

}

@Test(groups = { "SystemTesting","End2EndTesting" })

public void s3() {

System.out.println("S3 Test Cases ::: SystemTesting Group");

}

}

package Tutorial1;

import org.testng.annotations.Test;

public class Pune1 {

@Test(groups = { "SanityTesting","End2EndTesting" })

public void p1() {

System.out.println("P1 Test cases ::: SanityTesting group");

}

@Test(groups = { "RegressionTesting","End2EndTesting" },dependsOnGroups= {"SanityTesting"})

public void p2() {

System.out.println("P2 Test Cases ::: RegressionTesting Group");

}

@Test(groups = { "SystemTesting","End2EndTesting" })

public void p3() {

System.out.println("P3 Test Cases ::: SystemTesting Group");

}

}

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

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

<suite name="Suite">

<test name="Test">

<groups>

<run>

<include name="End2EndTesting"/>

</run>

</groups>

<classes>

<class name="Tutorial1.Sample1" />

<class name="Tutorial1.Demo1" />

<class name="Tutorial1.Pune1" />

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

**How to run fail Test cases again and again? VVVVVIMMMPPPPPPP**

there are 3 ways we can run failed test cases

1) by using testng-failed.xml file

2) by using IRetryAnalyzer interface

3) by using IAnnotationTransformar interface

**1) by using testng-failed.xml file**

- Go to test-output folder and then open a testng-failed.xml file and

then right click on xml file and run using TestNG Suite.

**2) by using IRetryAnalyzer interface**

- we have to implements IRetryAnalyzer interface by using implements keyword.

- we have to override the retry() method from IRetryAnalyzer interface.

- there we define how many times we have to run failed test cases.

syntax

public class Sample1 implements IRetryAnalyzer

{

int counter=0;//4

int maxRetryCounter=4;

public boolean retry(ITestResult result)

{

if(counter<maxRetryCounter)

{

counter++;

return true;

}

return false;

}

}

public class Demo1

{

@Test(retryAnalyzer=Sample1.class)

public void pqr()

{

Assert.assertEquals("Selenium", "Cucumber");

}

}

package Tutorial2;

import org.testng.IRetryAnalyzer;

import org.testng.ITestResult;

public class Sample1 implements IRetryAnalyzer{

int counter=0;//4

int maxRetryCounter=4;

@Override

public boolean retry(ITestResult result) {

if(counter<maxRetryCounter)

{

counter++;//4

return true;

}

return false;

}

}

package Tutorial2;

import org.testng.Assert;

import org.testng.annotations.Test;

public class Demo1 {

@Test

public void abc() {

System.out.println("ABC Test Cases");

Assert.assertEquals(12, 12);

}

@Test(retryAnalyzer=Sample1.class)

public void pqr() {

System.out.println("PQR Test Cases");//1 2 3 4 5

Assert.assertEquals("selenium", "cucumber");

}

@Test

public void xyz() {

System.out.println("XYZ Test Cases");

Assert.assertTrue(true);

}

}

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

**3) by using IAnnotationTransformar interface**

- we can run fail test cases again and again by implementing the IAnnotationTransformer interface using implements keyword.

- from IAnnotationTransformer interface we have to override the transform() method.

- inside this method we have to use setRetryAnanlyzer method by passing IRetryAnalyzer implemented class name.

syntax:

public class Pune2 implements IAnnotationTransformer

{

public void transform(ITestAnnotation annotation, Class testClass, Constructor testConstructor, Method testMethod)

{

annotation.setRetryAnalyzer(Sample2.class);

}

}

public class Sample2 implements IRetryAnalyzer

{

int counter=0;

int maxRetryCounter=4;

public boolean retry(ITestResult result)

{

if(counter<maxRetryCounter)

{

counter++;

return true;

}

return false;

}

}

XML

<suite name="suite name or project name">

<listeners>

<listener class-name="Tutorial2.Pune2"/>

</listeners>

<test name="test case name">

<classes>

<class name="Tutorial2.Demo2"/>

</classes>

</test>

</suite>

public class Demo2

{

@Test

public void abc()

{

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

}

@Test

public void pqr()

{

Assert.assertEquals("12", "12");

}

@Test

public void xyz()

{

Assert.assertEquals(123, 123);

}

}

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

package Tutorial2;

import org.testng.Assert;

import org.testng.annotations.Test;

public class Demo2 {

@Test

public void abc() {

System.out.println("ABC Test Cases");

Assert.assertEquals(12, 55555);

}

@Test

public void pqr() {

System.out.println("PQR Test Case");

Assert.assertEquals("selenium", "cucumber");

}

@Test

public void xyz() {

System.out.println("XYZ Test Case");

Assert.assertTrue(true);

}

}

package Tutorial2;

import org.testng.IRetryAnalyzer;

import org.testng.ITestResult;

public class Sample2 implements IRetryAnalyzer{

int counter=0;

int maxCounter=4;

@Override

public boolean retry(ITestResult result) {

if(counter<maxCounter)

{

counter++;

return true;

}

return false;

}

}

package Tutorial2;

import java.lang.reflect.Constructor;

import java.lang.reflect.Method;

import org.testng.IAnnotationTransformer;

import org.testng.annotations.ITestAnnotation;

public class Pune2 implements IAnnotationTransformer {

@Override

public void transform(ITestAnnotation annotation, Class testClass, Constructor testConstructor, Method testMethod) {

annotation.setRetryAnalyzer(Sample2.class);

}

}

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

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

<suite name="Suite">

<listeners>

<listener class-name="Tutorial2.Pune2"></listener>

</listeners>

<test name="Test">

<classes>

<class name="Tutorial2.Demo2"/>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

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

Implements IRetryAnalyzer interface

package Tutorial3;

public class Sample1 implements IRetryAnalyzer

{

int counter=0;

int maxCounter=10;

@Override

public boolean retry(ITestResult result)

{

if(counter<maxCounter)

{

counter++;

return true;

}

return false;

}

}

implements IAnnotationTransformer

public class Pune1 implements IAnnotationTransformer

{

@Override

public void transform(ITestAnnotation annotations, Class testClass, Constructor testConstructor, Method testMethod)

{

annotations.setRetryAnalzyer(Sample1.class);

}

}

public class Demo1

{

@Test

public void a1()

{

Assert.assertEquals(12,12);

}

}

XML

<suite name="Suite name">

<listeners>

<listener class-name="Tutorial3.Pune1"/>

</listeners>

<test>

<classes>

<class name="Tutorial3.Demo1"/>

</classes>

</test>

</suite>

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

package Tutorial3;

import org.testng.IRetryAnalyzer;

import org.testng.ITestResult;

public class Sample1 implements IRetryAnalyzer{

int counter=0;

int maxCounter=10;

@Override

public boolean retry(ITestResult result)

{

if(counter<maxCounter)

{

counter++;

return true;

}

return false;

}

}

package Tutorial3;

import java.lang.reflect.Constructor;

import java.lang.reflect.Method;

import org.testng.IAnnotationTransformer;

import org.testng.annotations.ITestAnnotation;

public class Pune1 implements IAnnotationTransformer{

@Override

public void transform(ITestAnnotation annotation, Class testClass, Constructor testConstructor, Method testMethod) {

annotation.setRetryAnalyzer(Sample1.class);

}

}

package Tutorial3;

import org.testng.Assert;

import org.testng.annotations.Test;

public class Demo1 {

@Test

public void abc() {

System.out.println("ABC Test Cases");

Assert.assertEquals(12, 12);

}

@Test

public void pqr() {

System.out.println("PQR Test Cases");

Assert.assertEquals("selenium", "cucumber");

}

@Test

public void xyz() {

System.out.println("XYZ Test Cases");

Assert.assertFalse(false);

Assert.assertTrue(true);

}

}

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

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

<suite name="Suite">

<listeners>

<listener class-name="Tutorial3.Pune1"/>

</listeners>

<test name="Test">

<classes>

<class name="Tutorial3.Demo1" />

<class name="Tutorial3.Demo2"/>

</classes>

</test> <!--

Test -->

</suite> <!--

Suite -->

package Tutorial3;

import org.testng.Assert;

import org.testng.annotations.Test;

public class Demo2 {

@Test(priority = 1)

public void login()

{

System.out.println("Login Test cases");

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

}

@Test(priority = 2)

public void verifyTitle()

{

System.out.println("verify Title Test cases");

Assert.assertEquals("hometitle", "URL");

}

@Test(priority = 3)

public void verifyLogo()

{

System.out.println("verify Logo Test cases");

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

}

}

**Read the test data from Excel sheet**

Step 1: check file is present or not in desktop location or project location.

we have to create object of File class by passing file location.

File f =new File("file location");

use exist() method to verify the file is present or not

boolean a = f.exist();

System.out.println(a);

step 2: we read the file content by using FileInputStream class.

we have to create object of FileInputStream class by passing File instance

FileInputStream fis = new FileInputStream(f);

Step 3: we have 2 different types of excel sheet

I) .xlsx excel sheet

ii) .xls excel sheet.

if excel sheet is .xlsx excel sheet then we have to create object of XSSFWorkbook by passing FileInputStream instance

XSSFWorkbook is used to load all the excel sheet.

XSSFWorkbook workbook =new XSSFWorkbook(fis);

if excel sheet is .xls excel sheet then we have to create object of HSSFWorkbook by passing FileInputStream instance

HSSFWorkkbook workbook =new HSSFWorkkbook(fis);

step 4: focus on specific sheet using getSheetAt() method by passing sheet index.

XSSFSheet sheet = workbook.getSheetAt(1);

return type of getSheetAt() method is XSSFSheet

step 5:capture the values

String a = sheet.getRow(9).getCell(1).getStringCellValue();

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

**Note:**

**if u want to read the test data from excel sheet then we have to add apache POI dependency in pom.xml file**

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

package Tutorial4;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Demo1 {

public static void main(String[] args) throws IOException {

//step 1: check the file is present or not

//create object of File class by passing file location

File f =new File("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData.xlsx");

boolean a= f.exists();

System.out.println(a);

//step 2: read the file content

//create object of FileInputStream by passing file instance

FileInputStream fis =new FileInputStream(f);

//step 3: load all work sheets

//create object of XSSFWorkbook by passing FileInputStream instance

XSSFWorkbook workbook =new XSSFWorkbook(fis);

//step 4: focus on specific sheet by using getSheetAt() method

XSSFSheet sheet = workbook.getSheetAt(0);

String b = sheet.getRow(8).getCell(4).getStringCellValue();

System.out.println(b);

}

}

package Tutorial4;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class Demo2 {

public static void main(String[] args) throws IOException {

File f =new File("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData.xlsx");

System.out.println(f.exists());

FileInputStream fis =new FileInputStream(f);

XSSFWorkbook workbook =new XSSFWorkbook(fis);

XSSFSheet sheet = workbook.getSheetAt(2);

String b = sheet.getRow(0).getCell(0).getStringCellValue();

WebDriver driver =new ChromeDriver();

driver.get(b);

String c = sheet.getRow(1).getCell(0).getStringCellValue();

WebElement fnamewb = driver.findElement(By.name("firstName"));

fnamewb.sendKeys(c);

String d = sheet.getRow(3).getCell(0).getRawValue();

System.out.println(d);

WebElement phoneWb = driver.findElement(By.name("phone"));

phoneWb.sendKeys(d);

}

}

**Data Driven Testing**

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

**How to capture all values from Excel sheet and print in console**

Step 1: check file is present or not by creating object of File class by passing location of file

File f =new File("file location");

step 2: read all file content by creating object FileInputStream by passing File class instance

FileInputStream fis =new FileInputStream(f);

step 3: load all work sheet by creating object of XSSFWorkbook by passing FileInputStream instance

XSSFWorkbook workbook =new XSSFWorkbook(fis);

step 4: focus on specific sheet by using getSheetAt() method and passing sheet index.

XSSFSheet sheet = workbook.getSheetAt(0);

step 5: get total count of rows using getLastRowNum() method

int rows = sheet.getLastRowNum()+1; //15

step 6: capture all first columns values

//iterate the rows

for(int i=0; i<rows; i++)

{

String a = sheet.getRow(i).getCell(0).getStringCellValue();

System.out.println(a);//firstname3

}

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

package Tutorial5;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Demo1 {

public static void main(String[] args) throws IOException {

// Step 1: check file is present or not by creating object of File class and by

// passing file location

File f = new File("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData.xlsx");

// step 2: read all file content by creating object of FileInputStream class and

// by passing File instance

FileInputStream fis = new FileInputStream(f);

// step 3: load all work sheet by creating object of XSSFWorkbook by passing

// FileInputStream instance

XSSFWorkbook workbook = new XSSFWorkbook(fis);

// step 4: focus on specific sheet using getSheetAt() method and by passing

// sheet index

XSSFSheet sheet = workbook.getSheetAt(2);

// step 5: count total numbers rows from sheet

int rows = sheet.getLastRowNum() + 1;

// step 6:capture all first column values

// step 6a) iterate all rows

for (int i = 0; i < rows; i++) {

String a = sheet.getRow(i).getCell(0).getStringCellValue();

System.out.println(a);

}

}

}

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

public class Demo1

{

public static XSSFWorkbook workbook;

public Demo1(String excelPath)

{

File f =new File(excelPath);

FileInputStream fis =new FileInputStream(f);

workbook =new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum()+1;

return rows;

}

public String getSpecificSheetData(int sheetIndex, int row, int cell)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

String data = sheet.getRow(row).getCell(cell).getStringCellValue();

return data;

}

}

package Tutorial5;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Demo2 {

public static XSSFWorkbook workbook;

public Demo2(String excelPath) throws IOException

{

File f =new File(excelPath);

FileInputStream fis =new FileInputStream(f);

workbook =new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum()+1;

return rows;

}

public String getSheetTestData(int sheetIndex, int row, int cell)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

String data = sheet.getRow(row).getCell(cell).getStringCellValue();

return data;

}

}

package Tutorial5;

import java.io.IOException;

public class Sample1 {

public static void main(String[] args) throws IOException {

Demo2 obj = new Demo2("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData.xlsx");

int rows = obj.getTotalRowCount(1);//

for(int i=0;i<rows;i++)

{

String a = obj.getSheetTestData(1, i, 0);

System.out.print(a + " \t");

String b = obj.getSheetTestData(1, i, 1);

System.out.print(b + " \t");

String c = obj.getSheetTestData(1, i, 2);

System.out.println(c);

}

}

}

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

package Tutorial5;

import java.io.IOException;

public class Sample2 {

public static void main(String[] args) throws IOException {

Demo2 obj =new Demo2("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData.xlsx");

int rows = obj.getTotalRowCount(4);

for(int i=0;i<rows;i++)

{

String a = obj.getSheetTestData(4, i, 0);

System.out.print(a +" \t");

String b = obj.getSheetTestData(4, i, 1);

System.out.print(b +" \t");

String c =obj.getSheetTestData(4, i, 2);

System.out.print(c + "\t");

String d = obj.getSheetTestData(4, i, 3);

System.out.print(d + "\t");

String e = obj.getSheetTestData(4, i, 4);

System.out.print(e + "\t");

String f = obj.getSheetTestData(4, i, 5);

System.out.print(f + "\t");

String g = obj.getSheetTestData(4, i, 6);

System.out.println(g);

}

}

}

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

package Tutorial5;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Demo2 {

public static XSSFWorkbook workbook;

public Demo2(String excelPath) throws IOException {

File f = new File(excelPath);

FileInputStream fis = new FileInputStream(f);

workbook = new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex) {

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum() + 1;

return rows;

}

public int getTotalColumnCount(int sheetIndex) {

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int cells = sheet.getRow(0).getLastCellNum();

return cells;

}

public String getSheetTestData(int sheetIndex, int row, int cell) {

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

String data = sheet.getRow(row).getCell(cell).getStringCellValue();

return data;

}

}

package Tutorial5;

import java.io.IOException;

public class Sample2 {

public static void main(String[] args) throws IOException, InterruptedException {

Demo2 obj =new Demo2("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData.xlsx");

int rows = obj.getTotalRowCount(0);

int cells = obj.getTotalColumnCount(0);

for(int i=0;i<rows;i++)

{

for(int j=0;j<cells;j++)

{

String p = obj.getSheetTestData(0, i, j);

System.out.print(p + "\t");

}

System.out.println();

}

}

}

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

package Tutorial5;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Demo3 {

public static XSSFWorkbook workbook;

public Demo3(String excelPath) throws IOException {

File f = new File(excelPath);

FileInputStream fis = new FileInputStream(f);

workbook = new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum()+1;

return rows;

}

public int getTotalColumnCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int cells = sheet.getRow(0).getLastCellNum();

return cells;

}

public String getSheetTestData(int sheetIndex, int row, int cell)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

String data = sheet.getRow(row).getCell(cell).getStringCellValue();

return data;

}

}

Data Driven Testing

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

Capture all values from excel sheet using Reusable script code and print in console

public class Demo1

{

public static XSSFWorkbook workbook;

public Demo1(String ExcelPath)

{

File f =new File(ExcelPath);

FileInputStream fis =new FileInputStream(f);

workbook =new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum()+1;

return rows;

}

public int getTotalColumnCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int cells = sheet.getRow(0).getLastCellNum();

return cells;

}

public String getSheetTestData(int sheetIndex, int row, int cell)

{

//focus on specific sheet

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

//focus on specific cell

XSSFCell cells = sheet.getRow(row).getCell(cell);

//check which type value is present in cell

if(cells.getCellType() == XSSFCell.CELL\_TYPE\_STRING)

{

String data1 = cells.getStringCellValue();

return data1;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_NUMERIC)

{

String data2 =cells.getRawValue();

return data2;

}

return null;

}

}

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

package Tutorial6;

import java.io.File;

import java.io.FileInputStream;

import org.apache.poi.xssf.usermodel.XSSFCell;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Demo1 {

public static XSSFWorkbook workbook;

public Demo1(String excelPath) throws Exception {

// step 1:: check file is present or not by creating object of File class and by

// passing

// file location

File f = new File(excelPath);

// step 2:: read all file content by creating object of FileInputStream and by

// passing File class Instance

FileInputStream fis = new FileInputStream(f);

// step 3: load the all work sheet by creating object of XSSFWorkbook and by

// passing FileInputStream instance

workbook = new XSSFWorkbook(fis);

}

//create non static method with 1 int arguments and int return type

public int getTotalRowCount(int sheetIndex)

{

//step 1: focus on specific sheet

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

//step 2: capture the total rows in sheet

int rows = sheet.getLastRowNum()+1;

//step 3: return the capture rows

return rows;

}

//create non static method with 1 int arguments and int return type

public int getTotalColumnCount(int sheetIndex)

{

//step 1: focus on specific sheet

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

//step 2: capture total columns in sheet by focusing on row 1

int cells = sheet.getRow(0).getLastCellNum();

//step 3: return the capture cells or columns

return cells;

}

//create non static method with 3 int arguments and Object return type

public Object getSheetTestData(int sheetIndex, int row, int cell)

{

//step 1: focus on specific sheet

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

//step 2: focus on specific cell

XSSFCell cells = sheet.getRow(row).getCell(cell);

//Step 3: check which type of value is present inside the cell

//if cell is blank then return blank space

if(cells.getCellType()== XSSFCell.CELL\_TYPE\_BLANK)

{

return "";

}

//if cell value type is String then return the String value

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_STRING)

{

//Step 4: capture the String cell value

String a = cells.getStringCellValue();

return a;

}

//if cell value type is numeric then return the numeric value

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_NUMERIC)

{

//step 5: capture the numeric cell value

String b = cells.getRawValue();

return b;

}

//if cell value type is boolean then return the boolean value

else if(cells.getCellType()== XSSFCell.CELL\_TYPE\_BOOLEAN)

{

boolean c = cells.getBooleanCellValue();

return c;

}

//if cell value type is formula then return the formula value

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_FORMULA)

{

String d = cells.getCellFormula();

return d;

}

//if above values are not matches then return the null.

return null;

}

}

package Tutorial6;

public class Sample1 {

public static void main(String[] args) throws Exception {

Demo1 obj = new Demo1("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData1.xlsx");

//count total rows in sheet

int rows = obj.getTotalRowCount(0);

//count total columns in sheet

int columns = obj.getTotalColumnCount(0);

Object[][] data =new Object[rows][columns];

//iterate the rows

for(int i=0;i<rows;i++)

{

//iterate the columns

for(int j=0;j<columns;j++)

{

//capture all values one by one

data [i][j] = obj.getSheetTestData(0, i, j);

}

}

System.out.println(data[1][2]);

}

}

package Tutorial6;

public class Sample2 {

public static void main(String[] args) {

Object [][] abc =new Object[4][2];

abc[0][0]= "Username1";

abc[0][1]="Password1";

abc[1][0]="Username2";

abc[1][1]="Password2";

abc[2][0]="Username3";

abc[2][1]="Password3";

abc[3][0]=12345677;

abc[3][1]=909090900;

System.out.println(abc[1][1]);

}

}

===

public class ExcelReader

{

public static XSSFWorkbook workbook;

public ExcelReader(String excelPath)

{

File f =new File(excelPath);

FileInputStream fis =new FileInputStream(f);

workbook =new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum()+1;

return rows;

}

public int getTotalColumnCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int cells = sheet.getRow(0).getLastCellNum();

return cells;

}

public Object getSheetTestData(int sheetIndex, int row, int cell)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

XSSFCell cells = sheet.getRow(row).getCell(cell);

if(cells.getCellType() == XSSFCell.CELL\_TYPE\_BLANK)

{

return "";

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_STRING)

{

String a= cells.getStringCellValue();

return a;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_NUMERIC)

{

String b = cells.getRawValue();

return b;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_FORMULA)

{

String c = cells.getFormulaValue();

return c;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_BOOLEAN)

{

boolean d = cells.getBooleanCellValue();

return d;

}

return null;

}

//create non static with Object[][] return type and 1 int arguments for sheet index

public Object [][] getAllSheetTestData(int sheetIndex )

{

//count total rows in sheet

int rows = getTotalRowCount(sheetIndex);

//count total columns in sheet

int cells = getTotalColumnCount(sheetIndex);

//create Object[][] array with same rows and same columns

Object[] [] data = new Object[rows] [cells];

//iterate all rows

for(int i=0;i <rows; i++)

{

//iterate the columns

for(int j=0; j< cells; j++)

{

//capture all values from excel sheet and store in Object [][] array.

data [i][j] =getSheetTestData(sheetIndex, i ,j);

}

}

return data;

}

}

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

package Tutorial6;

import java.io.File;

import java.io.FileInputStream;

import org.apache.poi.xssf.usermodel.XSSFCell;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelReader {

public static XSSFWorkbook workbook;

public ExcelReader(String excelPath) throws Exception {

File f = new File(excelPath);

FileInputStream fis = new FileInputStream(f);

workbook = new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex) {

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum() + 1;

return rows;

}

public int getTotalColumnCount(int sheetIndex) {

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int cells = sheet.getRow(0).getLastCellNum();

return cells;

}

public Object getSheetTestData(int sheetIndex, int row, int cell)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

XSSFCell cells = sheet.getRow(row).getCell(cell);

if(cells.getCellType() == XSSFCell.CELL\_TYPE\_BLANK)

{

return "";

}

else if (cells.getCellType() == XSSFCell.CELL\_TYPE\_STRING)

{

String a = cells.getStringCellValue();

return a;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_NUMERIC)

{

String b = cells.getRawValue();

return b;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_BOOLEAN)

{

boolean c = cells.getBooleanCellValue();

return c;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_FORMULA)

{

String d = cells.getCellFormula();

return d ;

}

return null;

}

public Object[][] getAllSheetTestData(int sheetIndex)

{

int rows = getTotalRowCount(sheetIndex);

int cells = getTotalColumnCount(sheetIndex);

Object[] [] data = new Object[rows][cells];

for(int i=0;i<rows;i++)

{

for(int j=0; j<cells;j++)

{

data[i][j] = getSheetTestData(sheetIndex, i, j);

}

}

return data;

}

}

package Tutorial6;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class Sample3 {

@Test(dataProvider="fbTestData")

public void abc(String firstName, String lastName, String username) throws InterruptedException {

WebDriver driver = new ChromeDriver();

driver.get("https://www.facebook.com/reg");

driver.findElement(By.name("firstname")).sendKeys(firstName);

driver.findElement(By.name("lastname")).sendKeys(lastName);

driver.findElement(By.name("reg\_email\_\_")).sendKeys(username);

Thread.sleep(4000);

driver.quit();

}

@DataProvider(name="fbTestData")

public Object[][] xyz() throws Exception

{

ExcelReader obj =new ExcelReader("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData1.xlsx");

Object[][] data = obj.getAllSheetTestData(0);

return data;

}

}

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

package Tutorial6;

import java.io.File;

import java.io.FileInputStream;

import org.apache.poi.xssf.usermodel.XSSFCell;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelReader {

public static XSSFWorkbook workbook;

public ExcelReader(String excelPath) throws Exception {

File f = new File(excelPath);

FileInputStream fis = new FileInputStream(f);

workbook = new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex) {

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum() + 1;

return rows;

}

public int getTotalColumnCount(int sheetIndex) {

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int cells = sheet.getRow(0).getLastCellNum();

return cells;

}

public Object getSheetTestData(int sheetIndex, int row, int cell)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

XSSFCell cells = sheet.getRow(row).getCell(cell);

if(cells.getCellType() == XSSFCell.CELL\_TYPE\_BLANK)

{

return "";

}

else if (cells.getCellType() == XSSFCell.CELL\_TYPE\_STRING)

{

String a = cells.getStringCellValue();

return a;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_NUMERIC)

{

String b = cells.getRawValue();

return b;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_BOOLEAN)

{

boolean c = cells.getBooleanCellValue();

return c;

}

else if(cells.getCellType() == XSSFCell.CELL\_TYPE\_FORMULA)

{

String d = cells.getCellFormula();

return d ;

}

return null;

}

public Object[][] getAllSheetTestData(int sheetIndex)

{

int rows = getTotalRowCount(sheetIndex);

int cells = getTotalColumnCount(sheetIndex);

Object[] [] data = new Object[rows][cells];

for(int i=0;i<rows;i++)

{

for(int j=0; j<cells;j++)

{

data[i][j] = getSheetTestData(sheetIndex, i, j);

}

}

return data;

}

}

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

package Tutorial6;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class Sample4 {

@Test(dataProvider = "testdata")

public void registerTestCase(String fName, String lName, String phone, String email, String city, String state)

throws InterruptedException {

WebDriver driver = new ChromeDriver();

driver.get("https://demo.guru99.com/test/newtours/register.php");

driver.findElement(By.name("firstName")).sendKeys(fName);

driver.findElement(By.name("lastName")).sendKeys(lName);

driver.findElement(By.name("phone")).sendKeys(phone);

driver.findElement(By.name("userName")).sendKeys(email);

driver.findElement(By.name("city")).sendKeys(city);

driver.findElement(By.name("state")).sendKeys(state);

Thread.sleep(7000);

driver.quit();

}

@DataProvider(name = "testdata")

public Object[][] getTestData() throws Exception {

ExcelReader obj = new ExcelReader("C:\\Users\\praf0\\OneDrive\\Desktop\\TestData1.xlsx");

Object[][] data = obj.getAllSheetTestData(0);

return data;

}

}

What is Data Driven Testing?

or

What is Data driven framework?

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

public class ExcelReader

{

//step 1: create constructor with 1 String arguments:

//step 1a) check file is present or not by creating object of File class and by passing file location

//step 1b) read all file content by creating object of FileInputStream class by passing File instance

//step 1c) load all work sheets by creating object of XSSFWorkbook class and by passing FileInputStream instance

//step 2: create non static method with int return type and 1 int arguments

//step 2a): focus on specific sheet using getSheetAt() method and by passing sheet index.

//step 2b) capture last row number of sheet using getLastRowNum() method

//step 3: create non static method with int return type and 1 int arguments

//step 3a) : focus on specific sheet using getSheetAt() method and by passing sheet index.

//step 3b) : capture the last column number using getLastCellNum() method

//step 4:create non static method with Object return type and 3 int arguments

//step 4a) : focus on specific sheet using getSheetAt() method and by passing sheet index.

//step 4b) : focus on specific cell using getRow().getCell() by passing row and column number

//step 4c) : check the cell is null or not by using if condition cells.getCellType()==XSSFCell.CELL\_TYPE\_BLANK

//step 4d) : check the cell value is string or not by using else if condition as cells.getCellType()==XSSFCell.CELL\_TYPE\_STRING

//step 4e) check cell value is numeric or not by using else if condition as cells.getCellType()==XSSFCell.CELL\_TYPE\_NUMERIC

//step 4f) : check the cell value is boolean or not by using else is condition as cells.getCellType()==XSSFCell.CELL\_TYPE\_BOOLEAN

//step 5g: check cell value is formula or not by using else if condition as cells.getCellType()==XSSFCell.CELL\_TYPE\_FORMULA

//step 5h): if cell type does not matches then return the null.

//step 5: create non static method with Object[][] return type and 1 int arguments

//step 5a): call getTotalRowCount() method by passing sheet index

//step 5b) call getTotalColumnCount() method by passing sheet index

//step 5c): create object of Object[][] array and by passing rows and columns

//step 5d) iterate all excel sheet rows

//step 5e) iterate all excel sheet columns

//step 5f) : capture all values from excel sheet by calling getSheetTestData() method by passing sheet index, rows and columns and

store all captured values to Object[][] arrays.

//step 5g) return the Object[][] array.

}

public class Demo1

{

//if excel sheet contains if we have 3 columns then we have to take 3 arguments for test cases.

@Test(dataProvider="testData")

public void abc(String firstName, String middleName , String lastName)

{

//selenium code

}

//Match dataProvider attribute name to @DataProvide annotation name.

@DataProvider(name="testData")

public Object[][] getTestData()

{

//Step 1: create object of ExcelReader class by passing location

//Step 2: call getAllSheetTestData() by using object name and return type is Object[][] array

//Step 3: return type Object[][] array.

}

}

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

public class ExcelReader

{

XSSFWorkbook workbook;

public ExcelReader(String excelPath)

{

File f =new File(excelPath);

FileInputStream fis =new FileInputStream(f);

workbook =new XSSFWorkbook(fis);

}

public int getTotalRowCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int rows = sheet.getLastRowNum()=1;

return rows;

//return workbook.getSheetAt(sheetIndex).getLastRowNum()+1;

}

public int getTotalColumnCount(int sheetIndex)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

int cells = sheet.getRow(0).getLastCellNum();

return cells;

//return workbook.getSheetAt(sheetIndex).getRow(0).getLastCellNum();

}

public Object getSheetTestData(int sheetIndex, int row, int cell)

{

XSSFSheet sheet = workbook.getSheetAt(sheetIndex);

XSSFCell cells = sheet.getRow(row).getCell(cell);

//XSSFCell cells = workbook.getSheetAt(sheetIndex).getRow(row).getCell(cell);

if(cells.getCellType()==XSSFCell.CELL\_TYPE\_BLANK)

{

return " ";

}

else if (cells.getCellType()==XSSFCell.CELL\_TYPE\_STRING)

{

return cells.getStringCellValue();

}

else if(cells.getCellType()==XSSFCell.CELL\_TYPE\_NUMERIC)

{

return cells.getRawValue();

}

else if(cells.getCellType()==XSSFCell.CELL\_TYPE\_BOOLEAN)

{

return cells.getBooleanCellValue();

}

else if(cells.getCellType()==XSSFCell.CELL\_TYPE\_FORMULA)

{

return cells.getFormulaValue();

}

return null;

}

public Object[][] getAllSheetTestData(int sheetIndex)

{

int rows = getTotalRowCount(sheetIndex);

int cells = getTotalColumnCount(sheetIndex);

Object [][] data = new Object[rows][cells];

for(int i=0;i<rows; i++)

{

for(int j=0;j<cells; j++)

{

data[i][j] =getSheetTestData(sheetIndex, i, j);

}

}

return data;

}

}

1) how to run methods parallel?

2) how to run classes parallel?

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

3) how to run test cases parallel? VVVVVIMMPPPPPPPPPP

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

1) how to run methods parallel?

<suite name="project name" parallel="methods">

<test name="test case name">

<classes>

<class name="packagename.classname"/>

</classes>

</test>

</suite>

package Tutorial7;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.Select;

import org.testng.annotations.Test;

public class Demo1 {

@Test

public void facebookTestCase1() throws InterruptedException {

WebDriver driver = new ChromeDriver();

driver.get("https://www.facebook.com/reg");

Thread.sleep(3000);

driver.findElement(By.name("firstname")).sendKeys("Pooja");

driver.findElement(By.name("lastname")).sendKeys("Mane");

driver.findElement(By.name("reg\_email\_\_")).sendKeys("pooja@gmail.com");

driver.findElement(By.name("reg\_passwd\_\_")).sendKeys("Pooja@123");

WebElement daywb = driver.findElement(By.name("birthday\_day"));

Select sel1 = new Select(daywb);

sel1.selectByVisibleText("21");

WebElement monthwb = driver.findElement(By.name("birthday\_month"));

Select sel2 = new Select(monthwb);

sel2.selectByVisibleText("Aug");

Thread.sleep(5000);

driver.quit();

}

@Test

public void guru99TestCase2() throws InterruptedException {

WebDriver driver = new ChromeDriver();

driver.get("https://demo.guru99.com/test/newtours/register.php");

Thread.sleep(3000);

driver.findElement(By.name("firstName")).sendKeys("rahul");

driver.findElement(By.name("lastName")).sendKeys("Shah");

driver.findElement(By.name("phone")).sendKeys("909090");

driver.findElement(By.name("userName")).sendKeys("rahul@gmail.com");

Thread.sleep(5000);

driver.quit();

}

}

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

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

<suite name="Suite" parallel="methods">

<test name="Test">

<classes>

<class name="Tutorial7.Demo1"/>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

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

2) how to run classes parallel

<suite name="project name" parallel="classes">

<test name="test case name">

<classes>

<class name="packagename.classname1"/>

<class name="packagename.classname2"/>

</classes>

</test>

</suite>

package Tutorial7;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.Select;

import org.testng.annotations.Test;

public class Demo2 {

@Test

public void facebookTestCase1() throws InterruptedException {

WebDriver driver = new ChromeDriver();

driver.get("https://www.facebook.com/reg");

Thread.sleep(3000);

driver.findElement(By.name("firstname")).sendKeys("Pooja");

driver.findElement(By.name("lastname")).sendKeys("Mane");

driver.findElement(By.name("reg\_email\_\_")).sendKeys("pooja@gmail.com");

driver.findElement(By.name("reg\_passwd\_\_")).sendKeys("Pooja@123");

WebElement daywb = driver.findElement(By.name("birthday\_day"));

Select sel1 = new Select(daywb);

sel1.selectByVisibleText("21");

WebElement monthwb = driver.findElement(By.name("birthday\_month"));

Select sel2 = new Select(monthwb);

sel2.selectByVisibleText("Aug");

Thread.sleep(5000);

driver.quit();

}

}

package Tutorial7;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.Select;

import org.testng.annotations.Test;

public class Sample2 {

@Test

public void facebookTestCase1() throws InterruptedException {

WebDriver driver = new ChromeDriver();

driver.get("https://www.facebook.com/reg");

Thread.sleep(3000);

driver.findElement(By.name("firstname")).sendKeys("Pooja");

driver.findElement(By.name("lastname")).sendKeys("Mane");

driver.findElement(By.name("reg\_email\_\_")).sendKeys("pooja@gmail.com");

driver.findElement(By.name("reg\_passwd\_\_")).sendKeys("Pooja@123");

WebElement daywb = driver.findElement(By.name("birthday\_day"));

Select sel1 = new Select(daywb);

sel1.selectByVisibleText("21");

WebElement monthwb = driver.findElement(By.name("birthday\_month"));

Select sel2 = new Select(monthwb);

sel2.selectByVisibleText("Aug");

Thread.sleep(5000);

driver.quit();

}

}

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

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

<suite name="Suite" parallel="classes">

<test name="Test">

<classes>

<class name="Tutorial7.Demo2"/>

<class name="Tutorial7.Sample2"></class>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

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

3) how to run test cases parallel VVVVVIMMPPPPPPPPPP

BaseLayer

BaseClass

-> static initialization method

--> create private static local thread of WebDriver

TestLayer

@BeforeTest --> call initialization() method

LoginPageTest ---->loginFunctionality test case

HomePageTest ----->

validate home page title

validate home page url

validate home page logo

ContactPageTest

-----> validate user is on contact page test case

-----> create user in contact page

----> delete user in contact page

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

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

<suite name="Suite" parallel="tests">

<test name="Test case 1">

<classes>

<class name="Tutorial7.LoginPageTest"/>

<class name="Tutorial7.HomePageTest"></class>

<class name="Tutorial7.ContactPageTest"></class>

</classes>

</test>

<test name="Test case 2">

<classes>

<class name="Tutorial7.LoginPageTest"/>

<class name="Tutorial7.HomePageTest"></class>

<class name="Tutorial7.ContactPageTest"></class>

</classes>

</test>

</suite>

package BaseLayer;

import java.time.Duration;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.edge.EdgeDriver;

public class BaseClass {

private static ThreadLocal<WebDriver> td = new ThreadLocal<WebDriver>();

public WebDriver getDriver() {

return td.get();

}

public static void initialization() {

WebDriver driver = new EdgeDriver();

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

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.get("https://ui.cogmento.com/");

td.set(driver);

}

}

package TestLayer;

import org.openqa.selenium.By;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import BaseLayer.BaseClass;

public class LoginPageTest extends BaseClass {

@BeforeTest

public void setUp() {

BaseClass.initialization();

}

@Test

public void loginFunctionality() throws InterruptedException {

Thread.sleep(5000);

getDriver().findElement(By.name("email")).sendKeys("prafulp1010@gmail.com");

getDriver().findElement(By.name("password")).sendKeys("Pr@ful0812");

Thread.sleep(5000);

getDriver().findElement(By.xpath("//div[text()='Login']")).click();

}

@AfterTest

public void tearDown() throws InterruptedException {

Thread.sleep(5000);

getDriver().quit();

}

}

package TestLayer;

import org.openqa.selenium.By;

import org.testng.Assert;

import org.testng.annotations.Test;

import BaseLayer.BaseClass;

public class HomePageTest extends BaseClass {

@Test(priority = 1)

public void validateHomePageTitle() {

String actualTitle = getDriver().getTitle();

Assert.assertEquals(actualTitle, "Cogmento CRM");

}

@Test(priority = 2)

public void validateHomePageUrl() throws InterruptedException {

String actualUrl = getDriver().getCurrentUrl();

boolean actualResult = actualUrl.contains("cogmento");

Assert.assertTrue(actualResult);

Thread.sleep(4000);

}

@Test(priority = 3)

public void validateHomePageLogo() {

boolean actualLogo = getDriver().findElement(By.xpath("//div[@class='header item']")).isDisplayed();

Assert.assertEquals(actualLogo, true);

}

}

package TestLayer;

import org.openqa.selenium.By;

import org.testng.Assert;

import org.testng.annotations.Test;

import BaseLayer.BaseClass;

public class ContactPageTest extends BaseClass {

@Test(priority = 1)

public void validateUserIsOnContactPage() throws InterruptedException {

getDriver().findElement(By.xpath("//a[@href='/contacts']")).click();

Thread.sleep(3000);

String actualUrl = getDriver().getCurrentUrl();

boolean actualResult = actualUrl.contains("contacts");

Assert.assertEquals(actualResult, true);

}

@Test(priority = 2)

public void createUserInContactPage() throws InterruptedException {

getDriver().findElement(By.xpath("//a[@href='/contacts/new']")).click();

getDriver().findElement(By.name("first\_name")).sendKeys("Rohit");

getDriver().findElement(By.name("last\_name")).sendKeys("Shah");

getDriver().findElement(By.name("value")).sendKeys("rohit@gmail.com");

Thread.sleep(7000);

getDriver().findElement(By.xpath("//button[text()='Save']")).click();

}

}

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

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

<suite name="Suite" parallel="tests">

<test name="Test case 1">

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

<test name="Test case 2">

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

</suite>

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

Cross browser Testing

package BaseLayer;

import java.time.Duration;

import org.openqa.selenium.Dimension;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.edge.EdgeDriver;

public class BaseClass {

private static ThreadLocal<WebDriver> td = new ThreadLocal<WebDriver>();

public WebDriver getDriver() {

return td.get();

}

public static void initialization(String browsername) {

WebDriver driver = null;

if (browsername.equalsIgnoreCase("chrome")) {

driver = new ChromeDriver();

td.set(driver);

} else if (browsername.equalsIgnoreCase("edge")) {

driver = new EdgeDriver();

td.set(driver);

} else if (browsername.equalsIgnoreCase("incognito")) {

ChromeOptions opt = new ChromeOptions();

opt.addArguments("--incognito");

driver = new ChromeDriver(opt);

td.set(driver);

} else {

System.out.println("Please Enter valid browser name");

}

Dimension d = new Dimension(800, 800);

driver.manage().window().setSize(d);

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(30));

driver.get("https://ui.cogmento.com/");

}

}

package TestLayer;

import org.openqa.selenium.By;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

import BaseLayer.BaseClass;

public class LoginPageTest extends BaseClass {

@Parameters({"browsername"})

@BeforeTest

public void setUp(String browserName)

{

BaseClass.initialization(browserName);

}

@Test

public void loginFunctionality() throws InterruptedException {

Thread.sleep(5000);

getDriver().findElement(By.name("email")).sendKeys("prafulp1010@gmail.com");

getDriver().findElement(By.name("password")).sendKeys("Pr@ful0812");

Thread.sleep(5000);

getDriver().findElement(By.xpath("//div[text()='Login']")).click();

}

@AfterTest

public void tearDown() throws InterruptedException {

Thread.sleep(5000);

getDriver().quit();

}

}

package TestLayer;

import org.openqa.selenium.By;

import org.testng.Assert;

import org.testng.annotations.Test;

import BaseLayer.BaseClass;

public class HomePageTest extends BaseClass {

@Test(priority = 1)

public void validateHomePageTitle() {

String actualTitle = getDriver().getTitle();

Assert.assertEquals(actualTitle, "Cogmento CRM");

}

@Test(priority = 2)

public void validateHomePageUrl() throws InterruptedException {

String actualUrl = getDriver().getCurrentUrl();

boolean actualResult = actualUrl.contains("cogmento");

Assert.assertTrue(actualResult);

Thread.sleep(4000);

}

@Test(priority = 3)

public void validateHomePageLogo() {

boolean actualLogo = getDriver().findElement(By.xpath("//div[@class='header item']")).isDisplayed();

Assert.assertEquals(actualLogo, true);

}

}

package TestLayer;

import org.openqa.selenium.By;

import org.testng.Assert;

import org.testng.annotations.Test;

import BaseLayer.BaseClass;

public class ContactPageTest extends BaseClass {

@Test(priority = 1)

public void validateUserIsOnContactPage() throws InterruptedException {

getDriver().findElement(By.xpath("//a[@href='/contacts']")).click();

Thread.sleep(3000);

String actualUrl = getDriver().getCurrentUrl();

boolean actualResult = actualUrl.contains("contacts");

Assert.assertEquals(actualResult, true);

}

@Test(priority = 2)

public void createUserInContactPage() throws InterruptedException {

getDriver().findElement(By.xpath("//a[@href='/contacts/new']")).click();

getDriver().findElement(By.name("first\_name")).sendKeys("Rohit");

getDriver().findElement(By.name("last\_name")).sendKeys("Shah");

getDriver().findElement(By.name("value")).sendKeys("rohit@gmail.com");

Thread.sleep(7000);

getDriver().findElement(By.xpath("//button[text()='Save']")).click();

}

}

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

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

<suite name="Suite" thread-count="10" parallel="tests">

<test name="End To End test case 1">

<parameter name="browsername" value="chrome"></parameter>

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

<test name="End To End test case 2">

<parameter name="browsername" value="edge"></parameter>

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

<test name="End To End test case 3">

<parameter name="browsername" value="incognito"></parameter>

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

<test name="End To End test case 4">

<parameter name="browsername" value="incognito"></parameter>

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

<test name="End To End test case 5">

<parameter name="browsername" value="chrome"></parameter>

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

<test name="End To End test case 6">

<parameter name="browsername" value="edge"></parameter>

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

<test name="End To End test case 7">

<parameter name="browsername" value="chrome"></parameter>

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

<test name="End To End test case 8">

<parameter name="browsername" value="chrome"></parameter>

<classes>

<class name="TestLayer.LoginPageTest" />

<class name="TestLayer.HomePageTest" />

<class name="TestLayer.ContactPageTest" />

</classes>

</test>

</suite>

**How to ignore exception in testng? VVVVIMMPPPPPPPP**

**or**

**How to handle exception in TestNG? VVVVIMMPPPPPPPP**

- we use expectedExceptions attribute to handle or ignore the exception in testng

syntax:

@Test(expectedExceptions=exceptionClassName.class)

public void abc()

{

//statement

}

e.g.

@Test(expectedExceptions=NullPointerException.class)

public void abc()

{

String a=null;

System.out.println(a.length()); //NullPointerException

}

package Tutorial1;

import org.testng.annotations.Test;

public class Demo1 {

@Test(expectedExceptions=NullPointerException.class)

public void abc()

{

System.out.println("Abc TC is started");

String a=null;

System.out.println(a.length());

System.out.println("Abc test cases is ended");

}

}

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

**Handle Selenium Exception**

@Test(expectedExceptions=NoSuchElementException.class)

public void xyz()

{

WebDriver driver =new ChromeDriver();

driver.get("https://www.facebook.com");

driver.findElement(By.name("abc2342342435")).click();

}

package Tutorial1;

import org.openqa.selenium.By;

import org.openqa.selenium.NoSuchElementException;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.Test;

public class Demo2 {

@Test(expectedExceptions = NoSuchElementException.class)

public void xyz() {

WebDriver driver = new ChromeDriver();

driver.get("https://www.facebook.com/reg");

driver.findElement(By.name("aqbc21233432425562")).click();

}

}

package Tutorial1;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.TimeoutException;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.annotations.Test;

public class Demo2 {

@Test(expectedExceptions=TimeoutException.class)

public void xyz() {

WebDriver driver = new ChromeDriver();

driver.get("https://www.facebook.com/reg");

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("aqbc21233432425562"))).click();

}

}

**how to write description for test cases?**

- we use description attribute to write the description for test cases

syntax:

@Test(description="test case description")

public void abc()

{

//statement

}

package Tutorial1;

import org.testng.annotations.Test;

public class Demo3 {

@Test(description = "Facebook Register Page test cases with validation")

public void validateRegister()

{

System.out.println("validateRegister Test cases");

}

}

**how to run same test cases multiple time? VVVVIMMPPPPPPPP**

- we use invocationCount attribute by passing count number

syntax:

@Test(invocationCount=10)

public void abc()

{

System.out.println("abc Test cases");

}

package Tutorial1;

import org.testng.annotations.Test;

public class Demo4 {

@Test(invocationCount = 10)

public void abc()

{

System.out.println("abc test cases");

}

}

**how to run """"same test cases multiple time within the time""""?**

- we use invocationTimeOut attribute by passing time in ms.

- if we are using invocationTimeOut attribute then compulsory we have to invocationCount attribute

- if we dont mention the invocationCount attribute then there is no use of using invocationTimeOut attribute.

syntax:

@Test(invocationCount=10, invocationTimeOut=5000)

public void abc()

{

//statement

}

package Tutorial1;

import org.testng.annotations.Test;

public class Demo4 {

@Test(invocationCount = 5, invocationTimeOut = 4000)

public void abc() throws InterruptedException

{

Thread.sleep(1000);

System.out.println("abc test cases");

}

}

package Tutorial1;

import org.testng.annotations.Test;

public class Demo4 {

@Test(invocationTimeOut = 1000)

public void abc() throws InterruptedException

{

Thread.sleep(2000);

System.out.println("abc test cases");

}

}

**how to allocate multiple thread for test cases? VVVVIMMPPPPPPPP**

- we use threadPoolSize attribute to provide the multiple thread for the same invocation count test cases

- compulsory we have to use invocationCount attribute to create multiple thread for same test cases.

- if we dont mention the invocationCount attribute then it ignore the threadPoolSize attribute

syntax;

@Test(invocationCount=5, threadPoolSize=3)

public void abc()

{

System.out.println("abc test cases");

}

package Tutorial1;

import org.testng.annotations.Test;

public class Demo5 {

@Test(invocationCount = 10, threadPoolSize = 5)

public void abc() throws InterruptedException

{

System.out.println("abc test cases");

Thread.sleep(3000);

}

}

package Tutorial1;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.Test;

public class Demo5 {

@Test(invocationCount = 10, threadPoolSize = 2)

public void abc() throws InterruptedException

{

WebDriver driver = new ChromeDriver();

driver.get("https://www.facebook.com");

Thread.sleep(7000);

driver.quit();

}

}

**how to run """""test cases within the time""""?**

- we use timeOut attribute to run the test cases within the time

syntax:

@Test(timeOut=2000)

public void abc()

{

Thread.sleep(3000);

System.out.println("abc test cases");

}

package Tutorial1;

import org.testng.annotations.Test;

public class Demo6 {

@Test(timeOut = 2000)

public void abc() throws InterruptedException

{

Thread.sleep(3000);

System.out.println("abc test cases");

}

}

package Tutorial1;

import org.testng.annotations.Test;

public class Demo6 {

@Test(invocationCount = 10 , timeOut = 2000)

public void abc() throws InterruptedException

{

Thread.sleep(1800);

System.out.println("abc test cases");

}

}

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

**diff between timeOut and invocationTimeOut**?

- timeOut is apply for single test cases

- and invocationTimeOut is apply for all invocationCount test cases.

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

Attribute in Test

1) priority attribute

2) dependsOnMethods attribute

3) dependsOnGroups attribute

4) enabled=false attribute

5) alwaysRun=true attribute

6) invocationCount=10 attribute

7) timeOut=time in milliseconds attribute

8) dataProvider= data provider name attribute

9) invocationTimeOut=time in milliseconds attribute

10) threadPoolSize=int number attribute

11) groups= group name attribute

12) expectedExceptions= exception name attribute

13) description= description of test cases attribute

14) retryAnalyzer= implemented class of IRetryAnalzyer interface attribute

Annotation in TestNG

1) @BeforeSuite

2) @BeforeTest

3) @BeforeClass

4) @BeforeMethod

5) @Test

6) @Ignore

7) @DataProvider

8) @Parameters

9) @AfterMethod

10) @AfterClass

11) @AfterTest

12) @AfterSuite

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

XML File structure with all XML tags

<suite name="suite name or project name" thread-count=5 parallel="tests">

<groups>

<run>

<include name="group name" />

</run>

</groups>

<listeners>

<listener class-name="packagename.classname"/>

</listeners>

<parameter name="parametername " value ="parameter value" />

<test name="test case name">

<classes>

<class name="packagename.classname"/>

</classes>

</test>

</suite>

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

XML File structure with all XML tags

<suite name="suite name or project name" thread-count=5 parallel="tests">

<listeners>

<listener class-name="packagename.classname"/>

</listeners>

<test name="test case name">

<parameter name="parametername " value ="parameter value" />

<groups>

<run>

<include name="group name" />

</run>

</groups>

<classes>

<class name="packagename.classname"/>

</classes>

</test>

</suite>

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

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

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

<suite name="Suite" thread-count="5" parallel="tests">

<groups>

<run>

<include name="regression"></include>

</run>

</groups>

<listeners>

<listener class-name="packagename.className"></listener>

</listeners>

<parameter name="browsername" value="chrome"></parameter>

<test name="Test">

<classes>

<class name="Tutorial1.Demo1" />

</classes>

</test>

</suite>

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

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

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

<suite name="Suite" thread-count="5" parallel="tests">

<listeners>

<listener class-name="packagename.className"></listener>

</listeners>

<test name="Test">

<parameter name="browsername" value="chrome"></parameter>

<groups>

<run>

<include name="regression"></include>

</run>

</groups>

<classes>

<class name="Tutorial1.Demo1" />

</classes>

</test>

</suite>

|  |  |
| --- | --- |
| **Hard Assert** | **Soft Assert** |
| Hard assert is also called as Assert | Soft Assert is also called as Verify |
| Hard assert it throws the exception immediately after assertion failure. | Soft Assert does not throws exception immediately during the executions |
| Hard assert it does not collect the exception | Soft assert it collect the exception during the executions. |
| Hard Assert all methods are static | Soft Assert all methods are non static. |
| Hard assert it throws exception immediately does not required to call any methods. | To throws in Soft Assert compulsory we have call assertAll() method. |
| After the assertion failure it will not going to execute remaining line of code | After assertion failure, it will execute remaining line of code. |

|  |  |
| --- | --- |
| **Before class** | **Before Method** |
| It is pre condition for all the test cases in class | It is pre condition for each and every test cases in class |

|  |  |
| --- | --- |
| **Before Class** | **Before Test** |
| It is pre condition for all the test cases in class | It is pre condition for all test cases in classes tags in XML File. |