Optional parameters

**package** test;

**import** org.testng.annotations.Optional;

**import** org.testng.annotations.Parameters;

**import** org.testng.annotations.Test;

**public** **class** OptionalParamTest {

@Test

@Parameters({ "browser", "url" })

// I used Optional annotation with parameter declaration

**public** **void** testMethodOne(@Optional("firefox") String testParameters1,

@Optional("http://www.google.com") String testParameters2) {

System.***out***.println("Paramters one for test method one: " + testParameters1);

System.***out***.println("Paramters two for test method one: " + testParameters2);

}

@Test

@Parameters({ "username" })

**public** **void** testMethodTwo(@Optional("shekar") String name)

{

System.***out***.println("Paramters one for test method two: " + name);

}

}

OptionalParam.xml

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

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

<suite name="OptionalParamSuite">

<test name="OptionalParamTest">

<!--

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

-->

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

<classes>

<class name="test.OptionalParamTest" />

</classes>

</test>

</suite> <!-- OptionalParamSuite -->

Summary :

IF xml file contains value for the parameter than , value of the xml file will be printer when the test is executed .

If xml file does not contain value for the parameter than , optional value available in the class file will be executed

Using Data provider :

Important feature of test ng

It is used for data driven test

Same test will run multiple times with different sets of data

**package** test;

**import** org.testng.annotations.DataProvider;

**import** org.testng.annotations.Test;

**public** **class** SampleDataProvider {

@DataProvider(name="browserdata")

**public** Object[][] dataProviderMethod()

{

**return** **new** Object[][] {{"chrome"},{"firefox"},{"safari"},{"ie"}};

}

@Test(dataProvider="browserdata")

**public** **void** testMethod(String browserdetails) {

System.***out***.println("Brwoser executed :" + browserdetails);

}

}

DataProvider.xml

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

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

<suite name="DataProviderSuite">

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

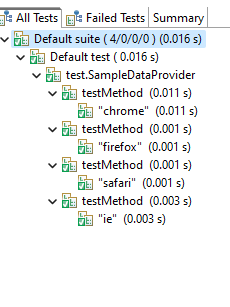
<classes>

<class name="test.SampleDataProvider"/>

</classes>

</test> <!-- DataProviderTest -->

</suite> <!-- DataProviderSuite -->



Summary:

Execution of test method is dependent on the number of datasets passed by the data provider method ({{"chrome"},{"firefox"},{"safari"},{"ie"}};)

Since we have 4 sets of values , the test method is executed 4 times .

Data provider method returns data in the form of double array of object class

First array represents data set .

Second array represent parameter values

Dependency test :

Testng allows a test method to depend on a single test method or group of test methods

**package** test;

**import** org.testng.Assert;

**import** org.testng.annotations.Test;

**public** **class** DependencyTest {

@Test(dependsOnMethods= {"Login"})

**public** **void** FundTransfer() {

System.***out***.println("Fund Transfer method called");

}

@Test

**public** **void** Login() {

System.***out***.println("Login method called");

//Assert.assertTrue(true);

Assert.*assertTrue*(**false**);

}

}

Dependency.xml

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

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

<suite name="DependencySuite">

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

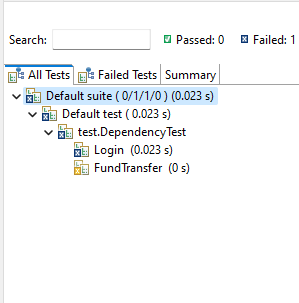
<classes>

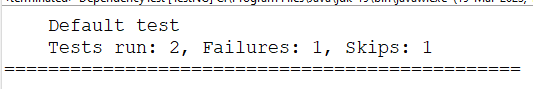
<class name="test.DependencyTest"/>

</classes>

</test> <!-- TDependencyest -->

</suite> <!-- DependencySuite -->





Summary : Fund transfer testcase will execute only when login test case is passed , because fund transfer test case has an attribute dependsOnMethods

Which is dependent of login test case result.

Parallelism or Multithreading :

Ability to execute multiple test cases simultaneously

Test NG allows test to run in parallel .

Advantage : it mainly reduces execution time

100 test cases -- 3 hrs -- serial order

100 test cases - 1 hrs ( parallelly 3 test cases ) – parallel execution

ParallelTestOne.java

**package** test;

**import** org.testng.annotations.Test;

**public** **class** ParallelTestOne {

@Test

**public** **void** testMethodOne() {

@SuppressWarnings("deprecation")

**long** id=Thread.*currentThread*().~~getId~~();

System.***out***.println("Class 1 : Thread id of method one is :"+id);

}

@Test

**public** **void** testMethodtwo() {

@SuppressWarnings("deprecation")

**long** id=Thread.*currentThread*().~~getId~~();

System.***out***.println("Class 1 : Thread id of method two is :"+id);

}

}

Assertions:

Hard Assert: it stops program execution if the statement with hard assert fails.

Whole test case will fail if one hard asset fails

SoftAssert: next steps would be executed if the statement with soft assert fails.

Whole test case will fail if one hard asset fails

How to re run failed test cases in Test Ng

1. Execute your test cases
2. Right click on project folder
3. Click on Refresh
4. Navigate to Test output folder
5. Run testng-failed.xml to execute failed test cases