It is a Unit testing Framework

**Purpose:** Design test cases in systematic way. It has zero licence cost.

Its available in the form of Jars files.

Java Unit testing frame work. It works with only with java.

**Advantages:**

* Test cases
* Good Html reports
* Lot of annotations
* Sequence of test cases also with priority
* Dependency feature
* Grouping Also possible
* Data Provider
* It’s also call TDD(Test Driven Development) frame work.

**Installation Test-NG in Eclipse:**

* Get Url: <http://beust.com/eclipse>
* Goto Eclipse – help—new s/w
* Put above url
* Install test-ng.
* Create Project
* Add test-ng jars in java Build path

**Importance of Xml file:**

Test Suite 🡪 Test Folder (Module) 🡪 Test Cases

Testng.xml file

**Include and Exclude Mechanism**

<class name=*"example1.Practic2"*>

<methods>

<include name = *"maillinktext"*/>

</methods>

</class>

<class name=*"testNG\_Topics.Solutions"*>

<methods>

<exclude name=*"Supply"*/>

</methods>

</class>

**Executing test cases By using Name:**

In Testing Name Convention is important, when we are developing framework, it is easy to run/exclude particular test cases with same name.

**Syntax:**

<exclude name=”testcase name.\*/”>

**Exclude/include whole Package test cases:**

**Annotations:**

Preconditiond start with @Before

@BeforeSuite

@BeforeClass

@BeforeMethod

@BeforeTest

Test Cases Start With @Test

@Test

**Grouping:**

**It is used to group the test cases**

**@Test(prority=2, groups=”Title”)**

**@Test(priority=3, groups=”search”)**

**Here also we can use exclude and include by using “group name”**

**<exclude name = “group name”>**

**TESTNG ANNOTATION HELPER ATTRIBUTES:**

**i) Depends On Methods:**

**This dependency is used to execute the specified step first**

**@test(dependsonmethod={“logout”,”close browser”}**

**Publiv void login()**

**{**

**}**

**Here Login execute first then logout, close browser methods execute**

**ii) Enabled:**

**@Test(enabled= false)**

**{**

**}**

**iii) Timeout:**

**If any flow take long time to execute, we can use this for specified block**

**@Test(timeout=4000)**

**Public void()**

**{**

**}**

**Priority:**

**It is used to make test cases execute by priority wise**

**@Test(priority=1)**

**Parametarization in TestNg XML File:**

**@Parameters**

**This parameter we use for pass the parameters in suite level or test folder level in testing.xml file.**

**<parameters name = “url” Value =”www.freeCRM.com”>**

**Data Provider Annotations:**

|  |  |
| --- | --- |
| Username | Password |
| naveenk | test@123 |
| Naveen@K | Test456 |
| Naveen67 | testdata |

Here we have 2 Columns, 3 rows, so we have to pass this data into multi-dimensional array like:

Object[][] data = new Object[3][2] 🡪 here we mention 3 rows, 2 coloumns

data[0][0]= "Naveen45";

data[0][1]="testingcheck";

data[1][0]="naveenk";

data[1][1]="test456";

data[2][0]="naveenk";

data[2][1]="test@123";

**By Using excel Sheet:**

**TestNG Listners:**

To Implement Listneres we have to use “ITest Listeners” interface.

We have to Keep TestNG listeners in TestNGXml file (testng4(dataprovider)

We can get Failed test name by using “getname”;

Ex:

System.***out***.println("This is Failed test" +result.getName());

**Running Tests In Parallel :**

If we want to trigger all test scripts all together at a time. Its give performance issues.

If we have to do:

We have to trigger in TestNGXml file

<suite name=*"testNG\_Topics"* parallel=*"tests"* thread-count=*"2"* >

Prallel 🡪 Test Names

Thread-count 🡪 how many tests to run

R

**Retry Logic:**

We can use this method to re execute faild test cases again , how many time we want by give counter.

Ex:

Counter =0 ; It strt from 0

Counter =3; times to execute

We can Use this logic at 2 levels:

**1) @ Test level**

If we have more test cases we cant write logic (retryAnalyzer = testNG\_Topics.RetryLogic.**class**))at every @Test so we have to go with below one.

**2) Runtime (Use Transform Analyzer)**

To run for all test cases, we have to use in testNG xml file

**Asserstion:**

We have to Make Sure where we have to use soft or Hard assertions.

**i) Hard Assertion**

If any particular assertion is failed, immediately after that remaining script won’t get execute in that test script. Test case terminates immediately.

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

**ii) Soft Assertion:**

If any particular soft assertion test failed also it will execute remaining Scripts.

Create a object of soft assert from TestNG class. We can write comment in this.

softAssert.assertEquals(**true**, **false**,"Test case failed");

**To markup the failed test cases**

**softAssert.assertAll();**

**Post Conditions Start with @After**

@AfterMethod

@AfterTest

@AfterClass

@AfterSuite

**To Get Report:**

* After execute the code, go to project and refresh it,
* Then Test Output folder creates
* In that index.html file copy location of this file and paste in browser