FRAMEWORKS

Page objects :-

Within your web app's UI there are areas that your tests interact with. A Page object simply models these as objects within the test code.

Pageobject is a script framework which is used to reusable page objects instead of repeating.

Why page objects :-

This reduces the amount of duplicated code it means that if the UI changes, the fix need only be applied in one place.

Page object declaration :-

```
Public static By Ref = By.xpath("  ");
```

Calling page objects :-

```
driver.findElement(classname.ref);
```

*

Pagefactory :-

Pagefactory is a part of pageobject
It contains library class called pagefactor
which supports to convert simple objects
into testcode.

Pagefactory element declaration :-

@FindBy(name="firstname") WebElement
Firstname_EB;
// @FindBy method look for object at
webpage [This object only access using
pagefactory class].

@FindBy(tagName="a") List<WebElement>
page_links;
// @FindBy also look for group of objects
at webpage

@FindBy(name="Pass") @CacheLookup
WebElement password_EB;
// @CacheLookup :- once object is located
and it will not be searched over and
over again.

[Mostly it is useful in ajax web application]

Note: Some times we get StaleElement ReferenceException.

```
@FindBys
  ({
     @FindBy(tagName="input"),
     @FindBy(tagName="Select"),
     @FindBy(ClassName="_8esa"),
  }) List<WebElement> By-Group-objects;
```

// When Required Webelement objects need to match all of the given criteria.

```
@FindAll
  ({
     @FindBy(tagName="Select"),
     @FindBy(ClassName="_8esa"),
  }) List<WebElement> All-Group-objects;
```

// When Required WebElement objects need to match atleast one of the given criteria

Run page Factory elements: inorder to get access @FindBy method we should take help of page factory class.

```
ClassName Ref = PageFactory.initElements
      (driver, Element classname.class);
```

TESTNG

TestNG : Test next generation

Testng is a opensource thirdparty framework. It derived from junit and Nunit Framework.

* TestNG support Test driven developm-ent framework [TDD]
* Report and running tool. [Running testcase and organizing testcase]
* Inbuilt framework of [Datadriven and keyword driven].

TestNG Advantages :

1. More Annotations available, to organize class and method execution.
2. Support parameterization using Dataprovider annotation
3. Excellent result generation compare to junit
4. Easy way to execute testcase from suite files.

5. Userdefined reports can be generate.

Note: TestNG is not a default plugin for eclipse, need to download from Eclipse marketplaces.

Creating Testing class:

Right click on package → Testing → create Testing class

Enter class name ignore all annotations click on finish.

Run testing class:

Right click on testing class → Select Runas → click TestingTest.

Note: In Real environment we should execute all testing classes from XML suite [TestNG Suite] file.

Where to find test results:

After execution of TestNG class, Refresh Project once, it generate testout folder additionally under your project. Expand folder and watch result under "emailablereport" or "index"

* Both are HTML files we can open using web browser.

Reporting Syntax at TestNG :

Reporter.log("Provide result");

→ Only generate result in Html file [Emailable report]

Reporter.log("Provide result", true);

→ result display at emailable report. Also under console.

How to ~~rep~~ prioritize test methods with in testNG class :

```
Public class testng class
{
  @Test(priority=0)
  Public void testmethod1()
  {
  }
}
```

Note: TestNG execute methods in alphabetical order. In order to execute methods in your own order specified at testNG

class. We can set priority for method

* Priority index start from zero.

How to provide description for testNG method :-

```
@Test(description = "write description")
public void test()
{
    Reporter.log("method Y is executed");
}
```

Note :-

After Generating result, In test output folder, Under Index → Click show All pass methods → Along with method name description will be display.

Also display method description with in console

* Description is not mandatory

How to ignore method execution with in testNG class :-

```
@Test (enabled = false)
Public void m1()
{
    Reporter.log("Method");
}
```

Note :-

* Find result at index ignored methods
* By using <<exclude>> tag we can ignore method from suite file.

How to provide execution dependency :-

for single method :-

```
@Test(dependsonMethods = {"dependent method name"})
```

for multiple methods :-

```
@Test(dependsonMethods = ({"dependent m1", dependentm2"})
```

Note :- if dependent method fail, TestNG skip method execution instead of making test result as fail.

How to insert assert statement in testNG class :

Assert :

Assert is a validation statement, It abort the run on failure. In testng it it abort the @Test method execution on failure.

AssertTrue :- Boolean comparision

AssertEquals :- Any string or other datatype comparision

```
Assert.assertEquals("Act", "Exp");
Reporter.log("Match Found");
Assert.assertTrue(true);
Reporter.log("Expected result found")
```

Assert statement send report to EmailableReports on Failure. On success manually we need to define using Reporter .log option

TestNG Suite file Architecture :

```
<Suite name="Suitename" >
  <test name="test1 " >
    <classes>
      <class name="Packagename.classname"
        <methods>
          <include name="methodname ">
          </include>
          <exclude name="methodname">
          </exclude>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

@BeforeSuite : The annotated method will be run before all tests in this suite have run.

@Aftersuite : The annotated method will be run after all tests in this suite have run.

@BeforeTest : The annotated method will be run ~~after all tests in this suite have run~~. before any test method belonging

to the classes inside the <test> tag is run

@AfterTest : The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.

@BeforeClass : The annotated method will be run ~~before all~~ the first test method in the current class is invoked.

@AfterClass : The annotated method will be run after all the test methods in the current class have been run.

@BeforeMethod : The annotated method will run before each test method.

@AfterMethod : The annotated method will run after each test method.

How to assign groups to TestNG test:

```
@Test(groups = "groupname")
```

→ Test assigning to single groups

```
@Test(groups = {"group1", "group2"})
```

→ Test assigning to multiple groups

Test Suite to run Groups:

```
<groups>
    <run>
        <include name="G2"> </include>
    </run>
</groups>
```

How to pass parameter from XML file:

```
<Suite name = "Suite1">
 <test name="test1">
  <classes>
   <class name="PN.CN">
    <parameter name="url" value="http://fb.com">
    </parameter>
    <parameter name="uid" value="saitharun">
    </parameter>
    <parameter name="pwd" value="sai123">
    </parameter>
```

```
</class>
</classes>
</test>
</suite>
```

Note:

name="url" → name of parameter

value="http://fb.com → value of parameter

How to retrieve parameter from XML:

```
@Test
@parameters("parameter_name_from_XML")
public void TestMethod(String anyname)
{
    S.O.P(anyname);
}
```

How to retrieve multiple parameter from XML:

```
@Test
@parameters({"par1","par2"})
public void TestMethod(String par1, String par2)
{
    S.O.P(par1);
    S.O.P(par2);
}
```

Note:→ we can find parameter information at emailable report. TestNG generate input values at testoutput folder.

How to execute two are multiple tests with in suite :

```
<suite name="suite1">
  <test name="test1">
      // Declare what classes to execute
  </test>
  <test name="test2">
      // Declare what classes to execute
  </test>
</suite>
```

How to execute multiple tests parallel :

```
<suite name="suite1" parallel="tests">
    // Define multiple tests to execute
</suite>
```

parallel="methods" :

TestNG will run all your test methods in seperate threads. Dependent methods will also run in seperate threads but they will ~~report~~ respect the order that you specified.

parallel = "tests" :-

TestNG will run all the methods in the same <test> tag in the same thread, but each <test> tag will be in a seperate thread. This allows you to group all your classes that are not thread safe in the same <test> and guarantee they will all run in the same thread while taking advantage of testNG using as many threads as possible to run your tests.

Parallel = "classes" :

TestNG will run all the methods in the same class in the same thread, but each class will be run in a seperate thread.

Parallel = "Instances" :

TestNG will run all the methods in the same instance in the same thread, but two methods on two different Instances will be running in different threads

Dataprovider :

Dataprovider is a annotation in testNG framework, It organize input Data to Execute test methods

Advantages :

* Without Defining Loops we can Execute test with multiple set of data
* Instead of following Excel kind of datadriven framework, we can organize multiple set of data to testcases.
* Dataprovider generate input data in emailable report. where as excel doesn't
* Annotation allows to create testdata easy way. without writing programing

Creating Data provider :

Right click on package --> Create testng class → Enter classname → Select Dataprovider annotation & Click on finish.

How to maintain data Under dataprovider method:

```
@Dataprovider
public String[][] dp()
{
 String data[][]=new String[2][2];
 data[0][0]="qadarshan@gmail.com";
 data[0][1]="newpassword";
 data[1][0]="newuser@gmail.com";
 data[1][1]="Userpwd123";
 return data;
}
```

Note:- @Dataprovider annotation we define where we maintain testdata to parameterize

How to retrive data from Dataprovider method

```
@Test(dataProvider="methodname")
Public void f(String uid, String pwd)
{
  // Method parameters need to declare on
}  Data provide method
```

How to retrieve data from different class

```
@Test(dataprovider="methodname",
        dataproviderclass=classname.class)
pubic void(String var, String var)
{
  //Data provider method should declare
  with static access specifier inorder
  to use outside a class
}
```