**TestNG**

Unit Testing tool to run test cases

* Can run the test cases without main method in java by using @Test annotation
* TestNG XML can be achieved by right clicking on Package>TestNG>ConvertToXML
* There are two options when user right clicks on Package TestNG>Convert to XML and CreateTestNGClass
* On clicking Create TestNG class, user can create test NG class by selecting the annotations

**TestNG XML Format:**

<Suite name=”SuiteName”>

<Groups>

<run>

<include name=”Smoke”>

</run>

</groups>

<test>

<Classes>

<class name=”test”>

<methods>

<exclude name=”Method that needs to be excluded”>

</methods>

</class>

</classes>

</test>

<classes>

<class name=”someclass forallmethods”/>

</classes>

<test>

</Suite>

**Annotations**:

* BeforeSuits: Executes before the whole suite
* AfterSuite: Executes after the whole suite
* BeforeClass: Executes before every class in package
* AfterClass: Exceutes after every class in package
* BeforeMethod: Executes before each method in a class
* AfterMethod: Executes after each method in class
* Test: To execute the test cases
* BeforeTest: executes before each test. If there are 2 @Test, then this will execute first and picks the @Test again executes BeforeTest and picks @Test
* AfterTest: Executes after each test
* Beforegroups: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.
* AfterGroups: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.
* DataProviders: Marks a method as supplying data for a test method. The annotated method must return an Object[ ][ ], where each Object[ ] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.

Syntax-Add a method for data providers to use the data

@Dataproviders

Public Object getdata(){

Object[][] data=new Object[3][2]

data[0][0]="firsrusername";

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

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

data[1][1]="secondpassword";

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

data[2][1]="thirdpssword";

**return** data;

}

@Test(dataproviders=”getdata”) needs to be added for a test to execute with multiple data

@Test(dataProvider="getData")

**public** **void** plMobile(String username, String password){

System.***out***.println("This is personal loan mobile");

System.***out***.println(username);

System.***out***.println(password);

* Factory: Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ].
* Listeners: Defines listeners on a test class.
* Parameters: Describes how to pass parameters to a @Test method.

Syntax in XML-If parameters needs to be defined for all tests in a suite then this needs to be added just after suite and before test or if it is specific to a test, then this needs to be added after test and before classes

<parameter name=”URL” value=”test.com”>

Syntax in Method:

* @Parameters({“URL”})

@Test

Public void method1(String urlName){

Sysout(“Print ”+urlName)}

**Annotation Helpers:**

dependsOnMethods: Syntax-@Test(dependsOnMethods={“Method1”,”Method2”})

timeOuts: Syntax-@Test(timeOut=4000)

**Groups:**

This is to add tags for the test if it is smoke or regression etc

Syntax-@Test(groups={“Smoke”,Regression})

XML:

<Suite name=”SuiteName”>

<Groups>

<run>

<include name=”Smoke”>

</run>

</groups>

<test>

**Parallel attribute in TestNG:**

If all tests in the suite has to run at a time, then use parallel=”tests” thread-count=”5” inside the suite starting tag

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

If all classes in a test has to run at a time, then use parallel=”tests” thread-count=”5” inside the test starting tag

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

**Listeners:**

IListeners is the interface. Listeners listen to the tests.

<listeners>

<listener class-name="prep2\_Testng.Listeners"/>

</listeners>

Create a class to implement ITestListeners. Eg if a test fails, then to take a screenshot for every failed test case, we can use a method in Listeners and ITestResult result can give the name of the class where there is a failure if we use result.getName()

**public** **void** onTestFailure(ITestResult arg0) {

// **TODO** Auto-generated method stub

System.***out***.println(arg0.getName()+" is failed");

}

How to create Properties file to define global parameters:

@Test

**public** **void** Login() **throws** IOException {

Properties prop=**new** Properties();

FileInputStream fis=**new** FileInputStream("C:\\Users\\Phani\\Documents\\selenium\\Interviews\\src\\test\\java\\prep2\_Testng\\variables.properties");

prop.load(fis);

System.***out***.println("User name is "+prop.getProperty("UserName"));

}

}

Reate a file as prop.properties in a package