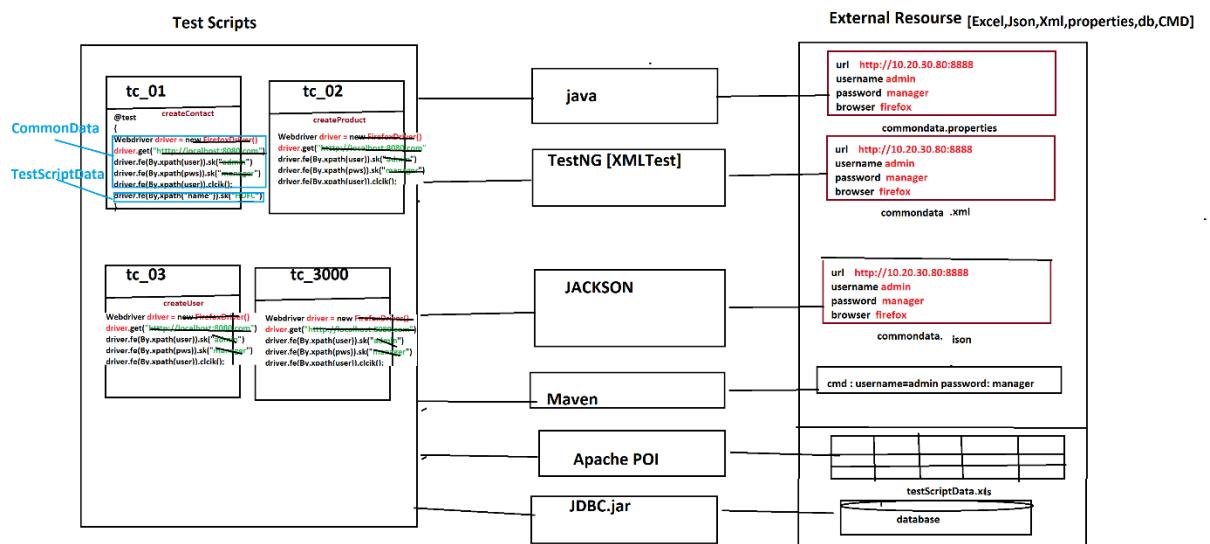


Data Driven testing

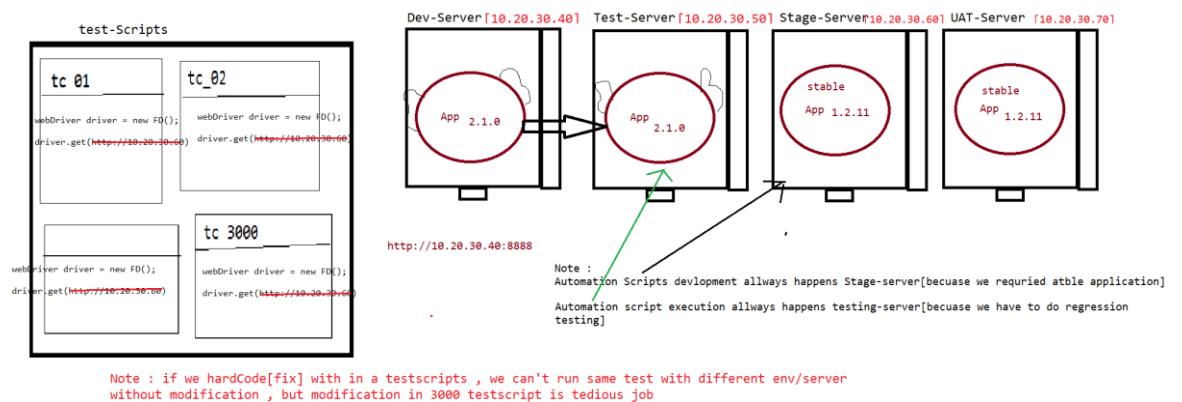
1. What is Data driven testing?

Read the data from external recourse & run the test is called Data driven testing (parameterization)



2. Why Data Driven testing?

As per the rule of the automation data shouldn't not hardcoded(fixed) with in a test scripts, because data modification & maintenance is tedious job when you want to run the test with different data, instead we should get the data from external resource like xlsx, .properties file , db , XML, JSON, CMD Line Data



3. What is Advantages of Data driven testing
 1. Maintenance of the test data is easy
 2. Modification of the test data in external recourse is easy
 3. Cross browser /platform testing is easy (means change the browser in property File)
 4. Running test scripts in different Environment is easy
 5. Running test scripts in different credentials is easy
 6. We can create the test data prior the Suite execution (we can also get the data from testData team)
 7. Rerunning same test Script with multiple time with different data is easy

Data driven testing from Properties File

1. What is Properties File?

Properties is java feature file where we can store the data in form of key & values pair,

Key & value data type should be always string .

```
url http://10.20.30.40:8888
browser firefox
username root
password manager
timeout 10
```

data.properties

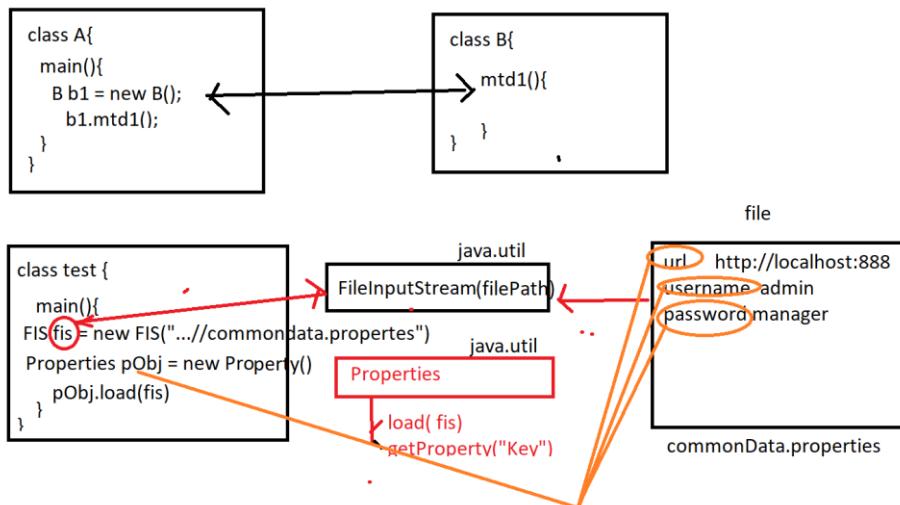
2. Why Property file ?

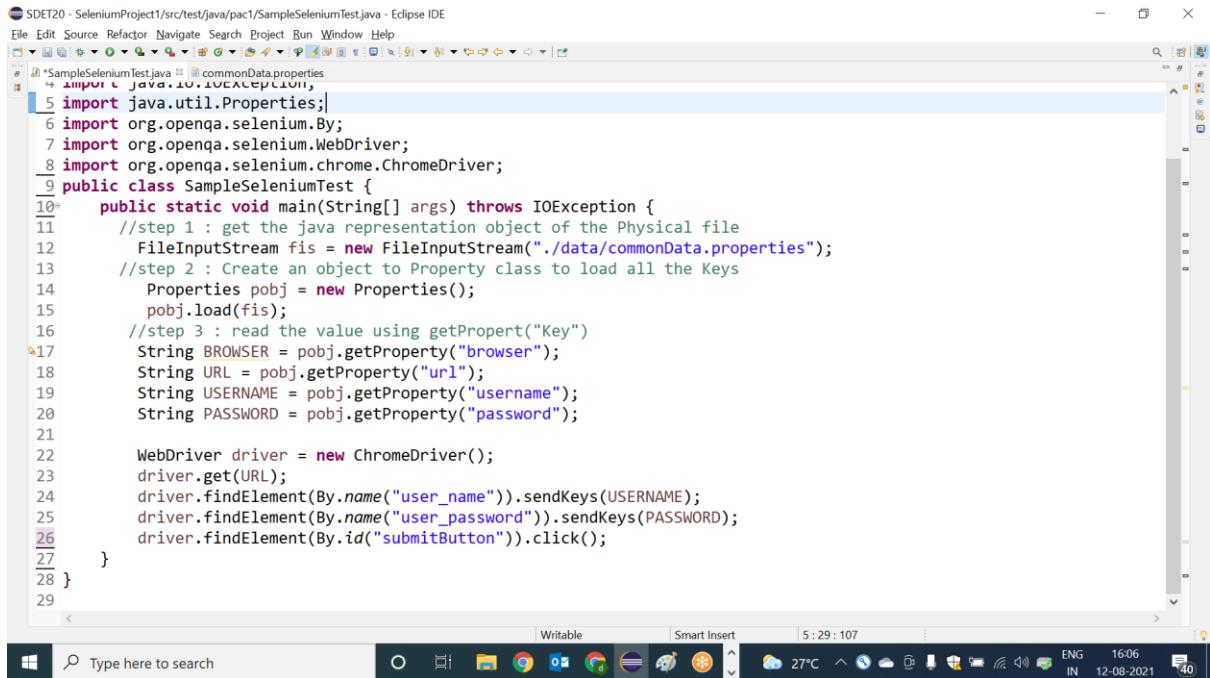
Property file is light weight & faster to read the data compare to any other file , & java as own Class to read the data from property

3. How to read data from properties File?

- Get the java representation Object of the Physical file using “FileInputStream”
- Create a Object of “Properties” class & load all the keys
- Read the data using getProperty(“Key”)

Note : properties file light weight & faster in execution compare to Excel



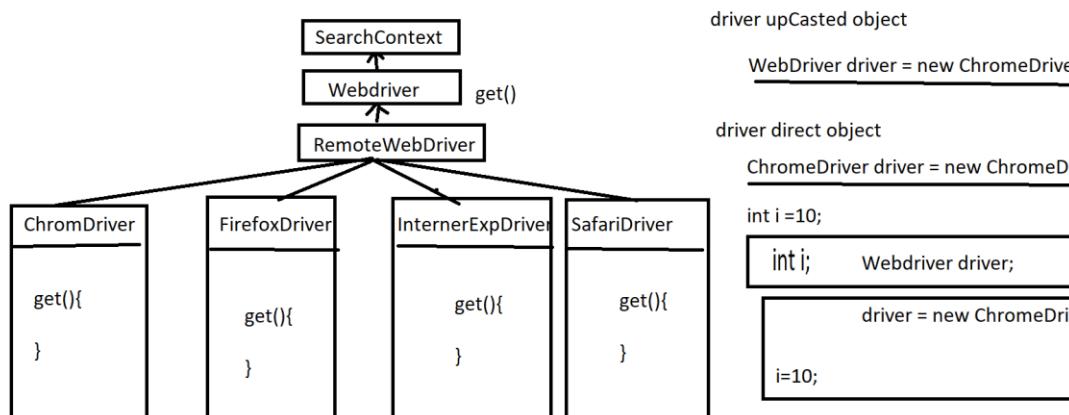


```

1 *SampleSeleniumTest.java 	commonData.properties
2 import java.io.IOException;
3 import java.util.Properties;
4 import org.openqa.selenium.By;
5 import org.openqa.selenium.WebDriver;
6 import org.openqa.selenium.chrome.ChromeDriver;
7 public class SampleSeleniumTest {
8     public static void main(String[] args) throws IOException {
9         //step 1 : get the java representation object of the Physical file
10        FileInputStream fis = new FileInputStream("./data/commonData.properties");
11        //step 2 : Create an object to Property class to load all the Keys
12        Properties pobj = new Properties();
13        pobj.load(fis);
14        //step 3 : read the value using getProperty("Key")
15        String BROWSER = pobj.getProperty("browser");
16        String URL = pobj.getProperty("url");
17        String USERNAME = pobj.getProperty("username");
18        String PASSWORD = pobj.getProperty("password");
19
20        WebDriver driver = new ChromeDriver();
21        driver.get(URL);
22        driver.findElement(By.name("user_name")).sendKeys(USERNAME);
23        driver.findElement(By.name("user_password")).sendKeys(PASSWORD);
24        driver.findElement(By.id("submitButton")).click();
25    }
26 }
27
28 }
29

```

How to use Browser data in Seleniumtest



EG : best example for run time Polymorphism , driver object behave differently in run time

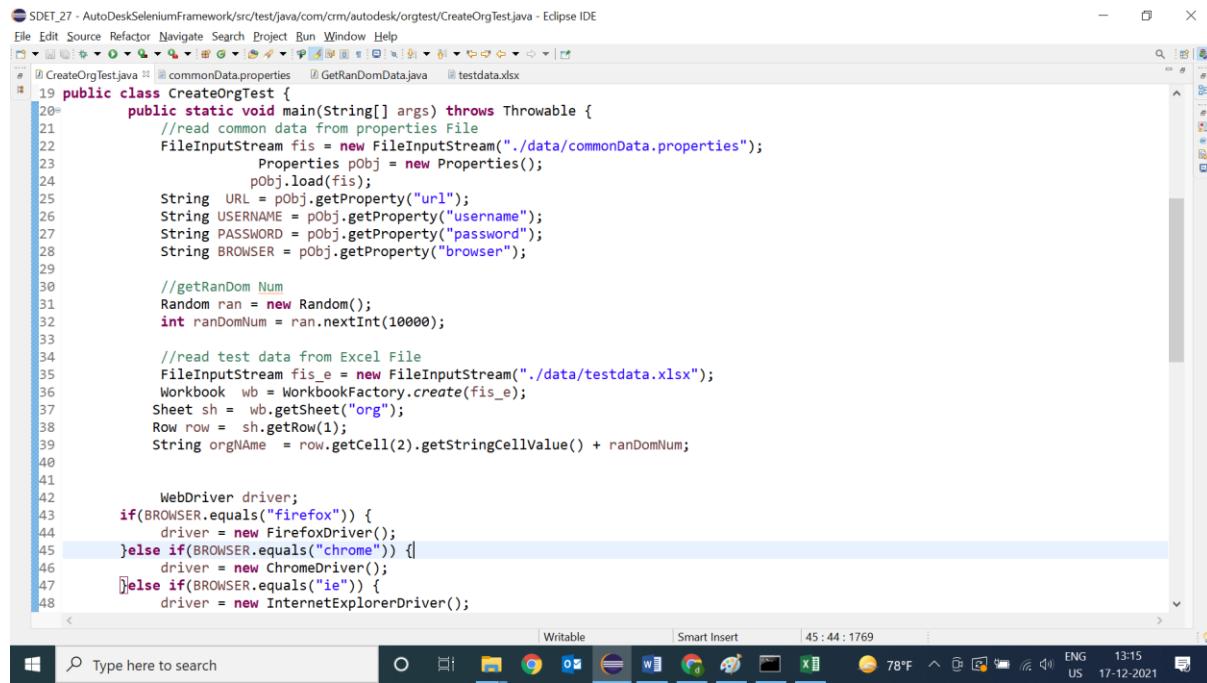
Assignment:

Test Case : CreateOrganization

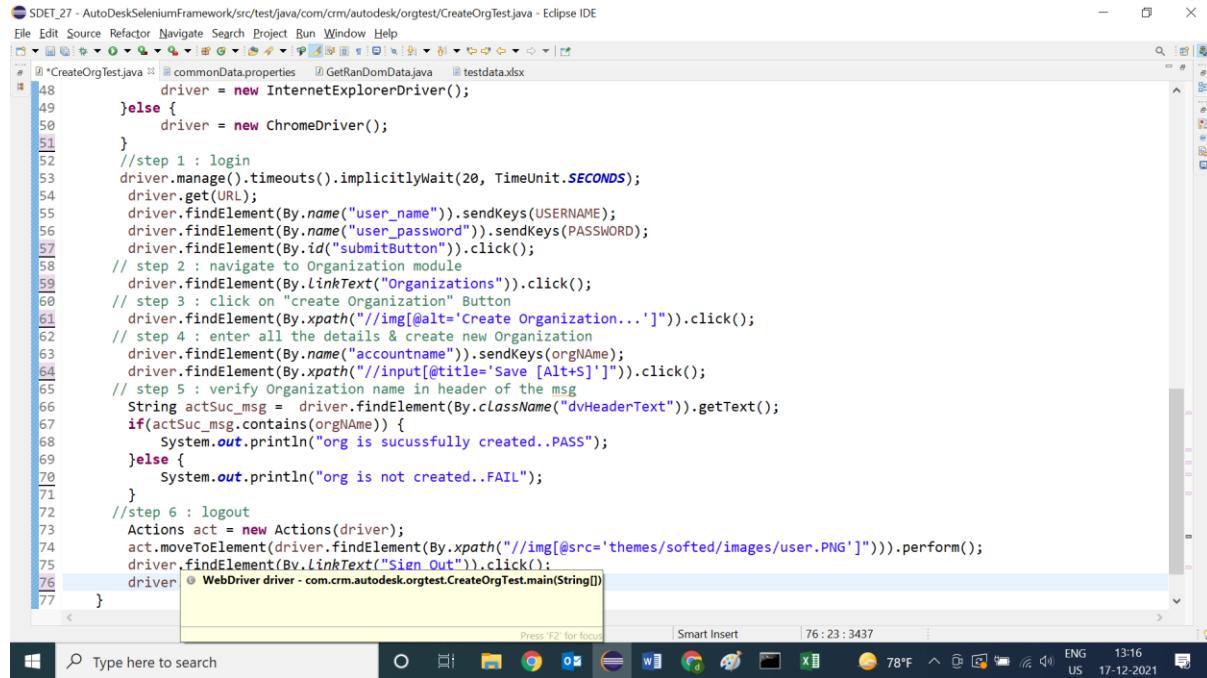
step 1 : login
 step 2 : navigate to Organization module
 step 3 : click on "create Organization" Button
 step 4 : enter all the details & create new Organization
 step 5 : verify Organization name in header of the msg
 step 6 : logout

Note :

1. Any data should not be hardcoded
 2. test should able to run in different browser with minimal changes
 3. test script data should get it from EXCEL sheet
-



```
SDET_27 - AutoDeskSeleniumFramework/src/test/java/com/crm/autodesk/orgtest/CreateOrgTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrgTestJava commonData.properties GetRandDomData.java testdata.xlsx
19 public class CreateOrgTest {
20     public static void main(String[] args) throws Throwable {
21         //read common data from properties File
22         FileInputStream fis = new FileInputStream("./data/commonData.properties");
23         Properties pobj = new Properties();
24         pobj.load(fis);
25         String URL = pobj.getProperty("url");
26         String USERNAME = pobj.getProperty("username");
27         String PASSWORD = pobj.getProperty("password");
28         String BROWSER = pobj.getProperty("browser");
29
30         //getRandDom Num
31         Random ran = new Random();
32         int ranDomNum = ran.nextInt(10000);
33
34         //read test data from Excel File
35         FileInputStream fis_e = new FileInputStream("./data/testdata.xlsx");
36         Workbook wb = WorkbookFactory.create(fis_e);
37         Sheet sh = wb.getSheet("org");
38         Row row = sh.getRow(1);
39         String orgName = row.getCell(2).getStringCellValue() + ranDomNum;
40
41
42         WebDriver driver;
43         if(BROWSER.equals("firefox")) {
44             driver = new FirefoxDriver();
45         }else if(BROWSER.equals("chrome")) {
46             driver = new ChromeDriver();
47         }else if(BROWSER.equals("ie")) {
48             driver = new InternetExplorerDriver();
49
50             driver = new InternetExplorerDriver();
51         }
52         //step 1 : login
53         driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
54         driver.get(URL);
55         driver.findElement(By.name("user_name")).sendKeys(USERNAME);
56         driver.findElement(By.name("user_password")).sendKeys(PASSWORD);
57         driver.findElement(By.id("submitButton")).click();
58         // step 2 : navigate to Organization module
59         driver.findElement(By.linkText("Organizations")).click();
60         // step 3 : click on "create Organization" Button
61         driver.findElement(By.xpath("//img[@alt='Create Organization...']")).click();
62         // step 4 : enter all the details & create new Organization
63         driver.findElement(By.name("accountname")).sendKeys(orgName);
64         driver.findElement(By.xpath("//input[@title='Save [Alt+S]']")).click();
65         // step 5 : verify Organization name in header of the msg
66         String actSuc_msg = driver.findElement(By.className("dvHeaderText")).getText();
67         if(actSuc_msg.contains(orgName)) {
68             System.out.println("org is sucessfully created..PASS");
69         }else {
70             System.out.println("org is not created..FAIL");
71         }
72         //step 6 : logout
73         Actions act = new Actions(driver);
74         act.moveToElement(driver.findElement(By.xpath("//img[@src='themes/softed/images/user.PNG']"))).perform();
75         driver.findElement(By.linkText("Sign Out")).click();
76         driver = WebDriver driver - com.crm.autodesk.orgtest.CreateOrgTest.main(String[])
77     }
}
```



```
SDET_27 - AutoDeskSeleniumFramework/src/test/java/com/crm/autodesk/orgtest/CreateOrgTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrgTestJava commonData.properties GetRandDomData.java testdata.xlsx
48         driver = new InternetExplorerDriver();
49     }else {
50         driver = new ChromeDriver();
51     }
52     //step 1 : login
53     driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
54     driver.get(URL);
55     driver.findElement(By.name("user_name")).sendKeys(USERNAME);
56     driver.findElement(By.name("user_password")).sendKeys(PASSWORD);
57     driver.findElement(By.id("submitButton")).click();
58     // step 2 : navigate to Organization module
59     driver.findElement(By.linkText("Organizations")).click();
60     // step 3 : click on "create Organization" Button
61     driver.findElement(By.xpath("//img[@alt='Create Organization...']")).click();
62     // step 4 : enter all the details & create new Organization
63     driver.findElement(By.name("accountname")).sendKeys(orgName);
64     driver.findElement(By.xpath("//input[@title='Save [Alt+S]']")).click();
65     // step 5 : verify Organization name in header of the msg
66     String actSuc_msg = driver.findElement(By.className("dvHeaderText")).getText();
67     if(actSuc_msg.contains(orgName)) {
68         System.out.println("org is sucessfully created..PASS");
69     }else {
70         System.out.println("org is not created..FAIL");
71     }
72     //step 6 : logout
73     Actions act = new Actions(driver);
74     act.moveToElement(driver.findElement(By.xpath("//img[@src='themes/softed/images/user.PNG']"))).perform();
75     driver.findElement(By.linkText("Sign Out")).click();
76     driver = WebDriver driver - com.crm.autodesk.orgtest.CreateOrgTest.main(String[])
77 }
```

Data driven testing from Excel File

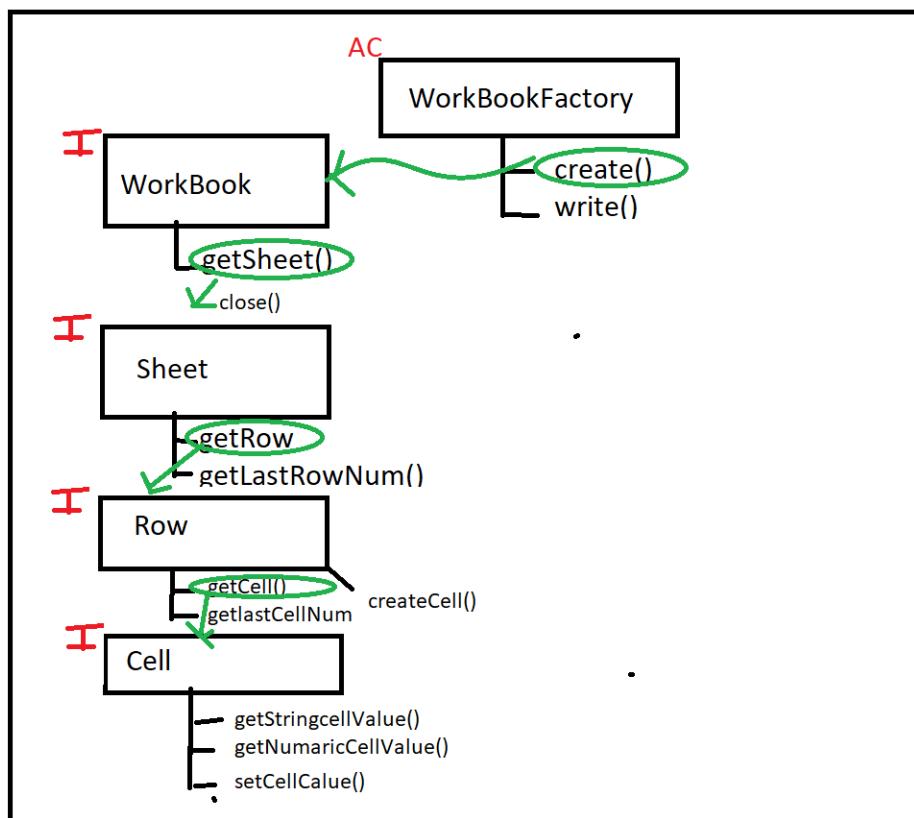
- Apache Poi is the open source libraries used to get & write data from all Microsoft documents like Excel , docx , ppt etc
- In real time most the company preferred the keep the test script data in Excel, because data will be in well-organized manner , so that modification & maintenance is easier.

Installation steps:

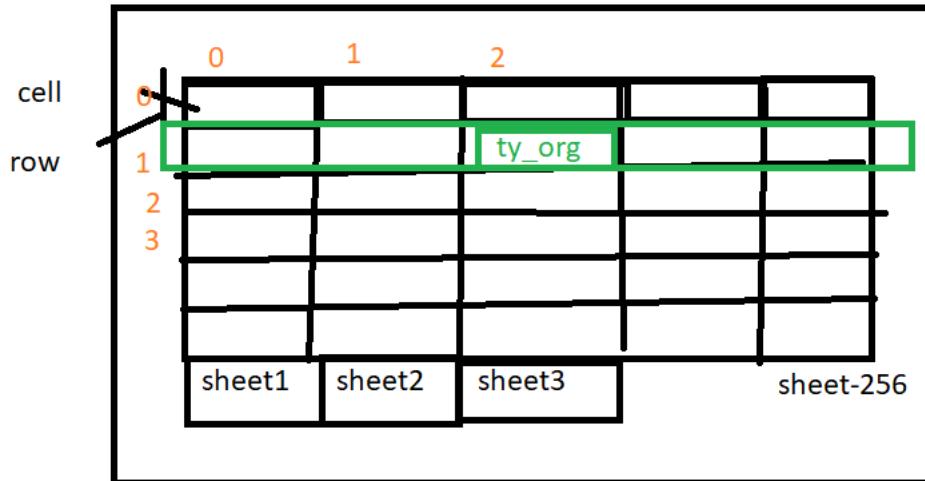
1. Go the Maven Project
2. Edit POM.xml file
3. Go to <https://mavenrepository.com>
4. Search for apache-POI
5. Copy the dependency
6. Add dependency inside the <dependencies> in POM.xml

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>4.0.0</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>4.0.0</version>
</dependency>
```

Class diagram of Apache POI



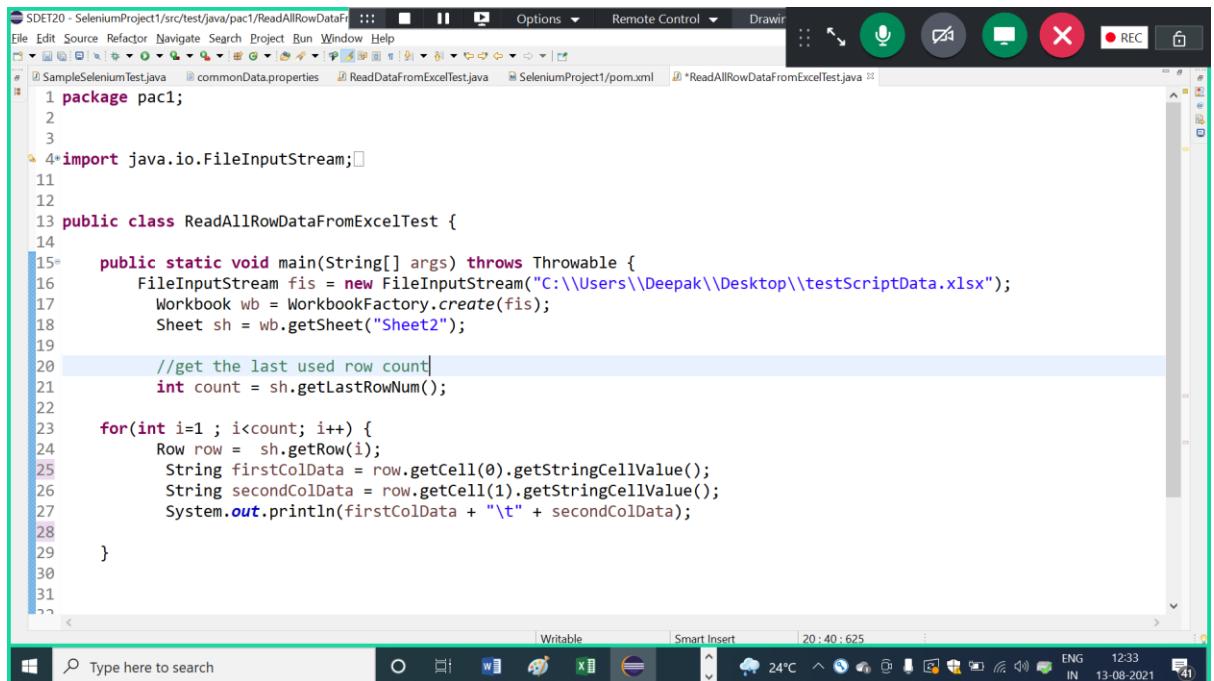
WorkBook



1. Program to read the data from WorkBook

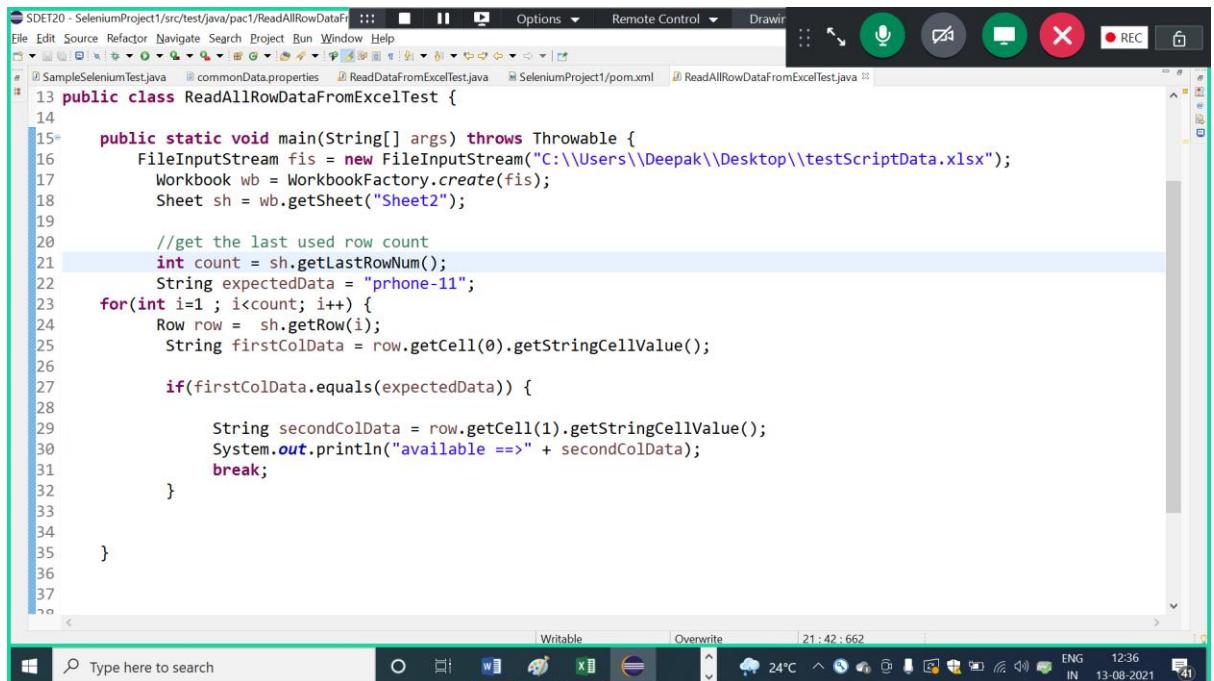
```
1 SDET20 - SeleniumProject1/src/test/java/pac1/ReadDataFromExcel.java :: Options Remote Control Drawin
2 File Edit Source Refactor Navigate Search Project Run Window Help
3 SampleSeleniumTest.java commonData.properties *ReadDataFromExcelTest.java SeleniumProject1/pom.xml
4
5 import org.apache.poi.ss.usermodel.Cell;
6 import org.apache.poi.ss.usermodel.Row;
7 import org.apache.poi.ss.usermodel.Sheet;
8 import org.apache.poi.ss.usermodel.Workbook;
9 import org.apache.poi.ss.usermodel.WorkbookFactory;
10
11
12 public class ReadDataFromExcelTest {
13
14
15    public static void main(String[] args) throws Throwable {
16        FileInputStream fis = new FileInputStream("C:\\\\Users\\\\Deepak\\\\Desktop\\\\testScriptData.xlsx");
17        //Step 1 : Open WorkBook in read mode
18        Workbook wb = WorkbookFactory.create(fis);
19        //Step 2 : get the control of the Sheet-1
20        Sheet sh = wb.getSheet("Sheet1");
21        //Step 3 : get the control of the 1 st Row
22        Row row = sh.getRow(1);
23        //Step 4 : get the control of the 2 nd Cell & copy the data
24        Cell cel = row.getCell(2);
25        String data = cel.getStringCellValue();
26        System.out.println(data);
27        //Step 5 :close the WorkBook
28        wb.close();
29    }
30 }
```

2. Program to read 1 st & 2 nd coloum data from all the Rows



```
1 package pac1;
2
3
4 import java.io.FileInputStream;
5
6
7 public class ReadAllRowDataFromExcelTest {
8
9     public static void main(String[] args) throws Throwable {
10         FileInputStream fis = new FileInputStream("C:\\Users\\Deepak\\Desktop\\testScriptData.xlsx");
11         Workbook wb = WorkbookFactory.create(fis);
12         Sheet sh = wb.getSheet("Sheet2");
13
14         //get the last used row count
15         int count = sh.getLastRowNum();
16
17         for(int i=1 ; i<count; i++) {
18             Row row = sh.getRow(i);
19             String firstColData = row.getCell(0).getStringCellValue();
20             String secondColData = row.getCell(1).getStringCellValue();
21             System.out.println(firstColData + "\t" + secondColData);
22         }
23
24     }
25
26 }
```

3. Write a program to find specific data from 1 st column , if data is available then read & dispaly next cell data



```
13 public class ReadAllRowDataFromExcelTest {
14
15     public static void main(String[] args) throws Throwable {
16         FileInputStream fis = new FileInputStream("C:\\Users\\Deepak\\Desktop\\testScriptData.xlsx");
17         Workbook wb = WorkbookFactory.create(fis);
18         Sheet sh = wb.getSheet("Sheet2");
19
20         //get the last used row count
21         int count = sh.getLastRowNum();
22         String expectedData = "prhone-11";
23
24         for(int i=1 ; i<count; i++) {
25             Row row = sh.getRow(i);
26             String firstColData = row.getCell(0).getStringCellValue();
27
28             if(firstColData.equals(expectedData)) {
29
30                 String secondColData = row.getCell(1).getStringCellValue();
31                 System.out.println("available ==>" + secondColData);
32                 break;
33             }
34
35         }
36
37 }
```

4. Program to write data back to WorkBook/ Excel

```
SDT20 - SeleniumProject1/src/test/java/pac1/WriteDataToExcelTest.java | Options | Remote Control | Drawin
File Edit Source Refactor Navigate Search Project Run Window Help
SampleSeleniumTest.java  SeleniumProject1/pom.xml  WriteDataToExcelTest.java
1 import org.apache.poi.ss.usermodel.Sheet;
2 import org.apache.poi.ss.usermodel.Workbook;
3 import org.apache.poi.ss.usermodel.WorkbookFactory;
4
5 public class WriteDataToExcelTest {
6
7     public static void main(String[] args) throws Throwable {
8
9         FileInputStream fis = new FileInputStream("C:\\\\Users\\\\Deepak\\\\Desktop\\\\testScriptData.xlsx");
10        //Open WorkBook in read mode
11        Workbook wb = WorkbookFactory.create(fis);
12        Sheet sh = wb.getSheet("Sheet1");
13        Row row = sh.getRow(1);
14        Cell cel = row.createCell(5);
15        cel.setCellValue("PASS");
16
17        //Open same WorkBook in write mode & save the File
18        FileOutputStream fos = new FileOutputStream("C:\\\\Users\\\\Deepak\\\\Desktop\\\\testScriptData.xlsx");
19        wb.write(fos);
20        wb.close();
21
22    }
23
24 }
25
26
27
28
29
30
31
32 }
33
34 }
```

A	B	C	D	E	F
1 TestID	TestName	orgName			PASS
2 tc_01	CreateOrg	TY_ORG			
4 TestID	TestName	orgName	orgName		
5 tc_02	CreateOrgWithType	Q_ORG	Energy		
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					
33					
34					

How to generate random number using java

The screenshot shows the Eclipse IDE interface with the title bar "SDET_27 - AutoDeskSeleniumFramework/src/test/java/practice/GetRanDomData.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc. The left sidebar shows project files: "CreateOrgTest.java", "commonData.properties", "GetRanDomData.java", and "testdata.xlsx". The main editor area contains the following Java code:

```
1 package practice;
2
3 import java.util.Random;
4
5 public class GetRanDomData {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9
10        Random ran = new Random();
11        int ranDomNum = ran.nextInt(10000);
12        System.out.println(ranDomNum);
13    }
14
15 }
16
```

The code generates a random integer between 0 and 9999 and prints it to the console. A TODO comment is present in the main method. The code is labeled as "Writable" and "Smart Insert" in the status bar at the bottom.

Data-Driven Testing from CommandLine – using Maven Parameters

Why Java Main method argument is String array ?

Because its always accept String array of data from the cammadline

4. How to get Maven Parameters(data like url , username , password) from the CMD line
 1. Download maven Command line plugin & set Environment variables

- Installation steps
1. go to google
 2. search download maven
 3. click on first link
 4. click on " apache-maven-3.6.3-bin.zip " beside "Binary zip archi"
 5. download -ZIP & extract
 6. get inside the folder & copy the "bin" path location
 7. Go to Local system Env varibale window & set below path

Environment Variables

Variable	Value
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_161
M2_HOME	D:\apache-maven-3.5.2-bin\apache-maven-3.5.2

Environment Variables

Variable	Value
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_161
M2_HOME	D:\apache-maven-3.5.2-bin\apache-maven-3.5.2
OneDrive	C:\Users\Deepak\OneDrive
Path	C:\Users\Deepak\AppData\Local\Programs\Python\Python38-32\Scripts\;C:\Users\Deepak\PyCharm 2020.2.1\bin;
PyCharm	C:\Program Files\JetBrains\PyCharm Community Edition 2020.2.1\bin\;
PyCharm Community Edition	C:\Program Files\JetBrains\PyCharm Community Edition 2020.2.1\b...
TEMP	C:\Users\Deepak\AppData\Local\Temp

Variable	Value
ANDROID_HOME	C:\Users\Deepak\AppData\Local\Android\Sdk
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\DriverData
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Program Files\Java\jdk1.8.0_161\bin;C:\Windows\system32;C:\Windows;C:\Windows\system32\Wbem;C:\Windows\system32\WindowsPowerShell\v1.0\;C:\Windows\system32\OpenSSH\;D:\apache-maven-3.5.2-bin\apache-maven-3.5.2\bin;C:\Users\Deepak\AppData\Local\Android\Sdk\platform-tools;C:\Users\Deepak\AppData\Local\Android\Sdk\tools
PATHEXT	.COM;.EXE;.BAT;.CMD;

8. go to command line , & execute below command to check maven installation

```
C:\Users\Deepak>mvn -version
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T13:28:13+05:30)
Maven home: D:\apache-maven-3.5.2-bin\apache-maven-3.5.2\bin\..
Java version: 1.8.0_161, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_161\jre
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

2. Create a Maven Project in Eclipse

3. Add testNG dependency in POM.xml

4. Create TestNg class

```
public class SampleTest {

    @Test
    Run | Debug
    public void createContactTest() {
        System.out.println("execute createContactTest");
        String URL = System.getProperty("url");
        String BROWSER = System.getProperty("browser");
        String USERNAME = System.getProperty("username");

        System.out.println(URL);
        System.out.println(BROWSER);
        System.out.println(USERNAME);

    }

}
```

5. Copy the location of the Maven Project

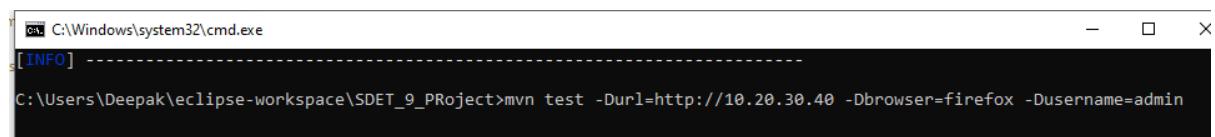
Project=> RightClick->Properties-> Recourse-> copy the -> location

6. Go to Cmd line

7. Go to the location of the Project

Cmd >cd C:\Users\Deepak\eclipse-workspace\SDET_9_Project

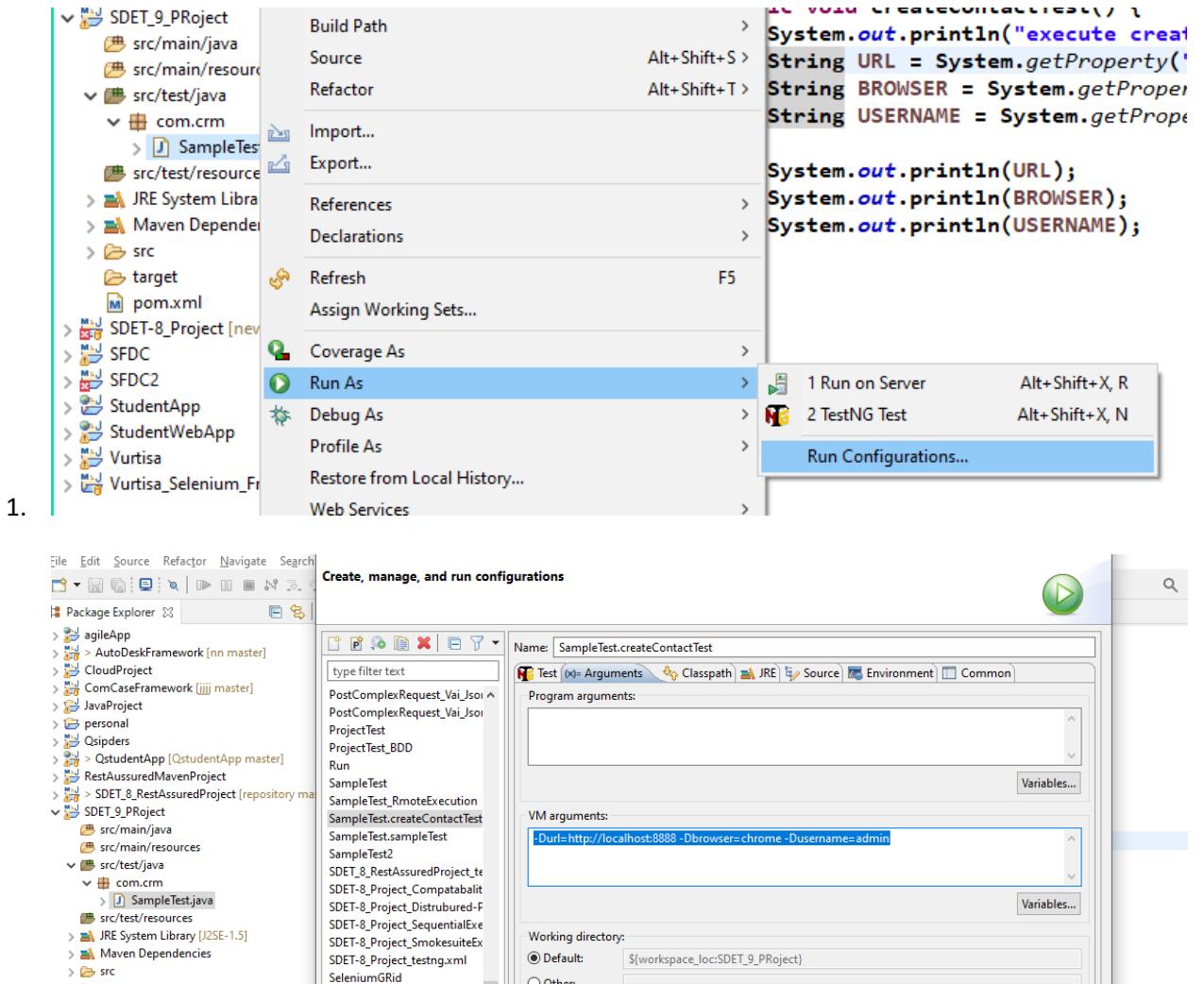
8. Execute below Command



The screenshot shows a Windows Command Prompt window titled 'cmd C:\Windows\system32\cmd.exe'. The window contains the following text:

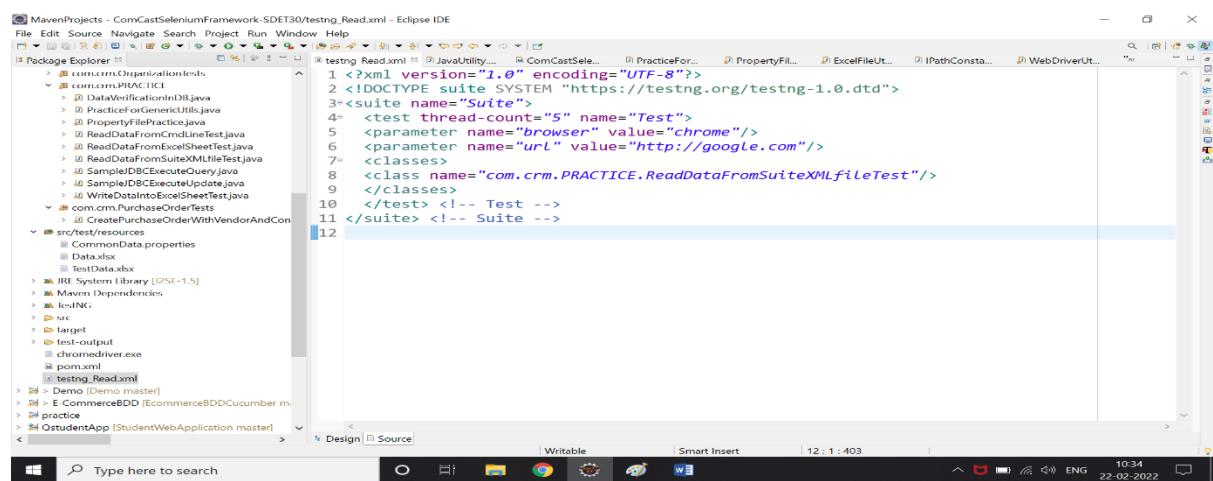
```
[INFO] --
C:\Users\Deepak\eclipse-workspace\SDET_9_PROJECT>mvn test -Durl=http://10.20.30.40 -Dbrowser=firefox -Dusername=admin
```

5. How to get Maven Parameters(data like url , username , password) from the Eclipse

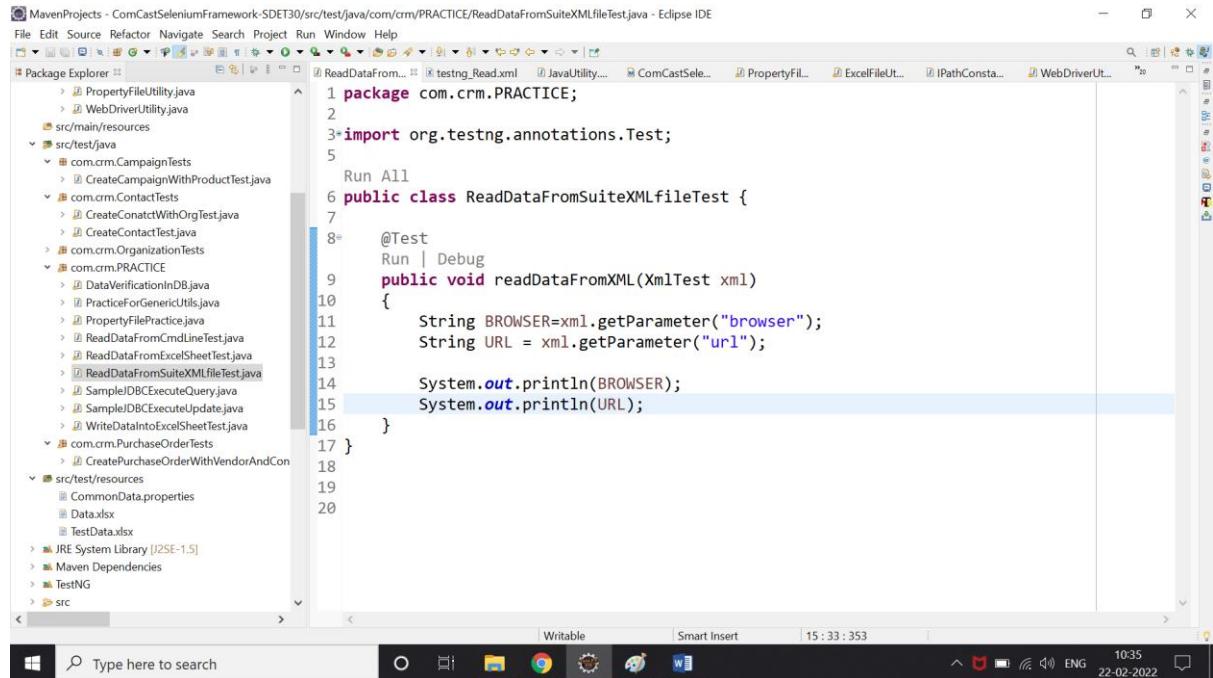


Data - Driven testing from Suite xml file

->Use parameter tag in suite xml file



→ Using `xmlTest` parameterise the method to read from suite xml file to the test script with the help of `getParameter()`.



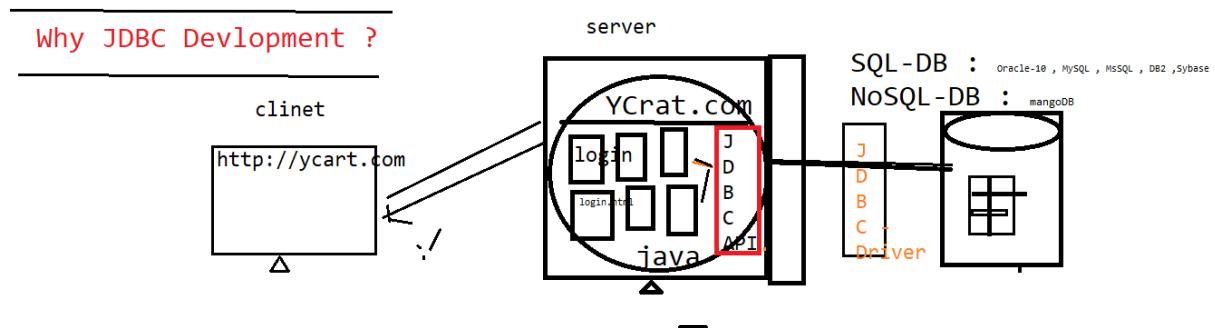
The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "MavenProjects - ComCastSeleniumFramework-SDT30". It includes packages like com.crm.CampaignTests, com.crm.ContactTests, com.crm.OrganizationTests, com.crm.PRACTICE, and com.crm.PurchaseOrderTests, along with various test classes and utility files.
- Editor:** Displays the Java code for "ReadDataFromSuiteXMLfileTest.java". The code imports org.testing.annotations.Test and defines a public class ReadDataFromSuiteXMLfileTest. It contains a single method readDataFromXML(XmlTest xml) that prints the browser and URL parameters using System.out.println.
- Bottom Status Bar:** Shows the status bar with "Writable", "Smart Insert", and system information including the date and time (22-02-2022, 10:35).

JDBC [Java Data base Connectivity]

What is JDBC ?

It's a J2EE API , which is used to connect to any data base from the java code



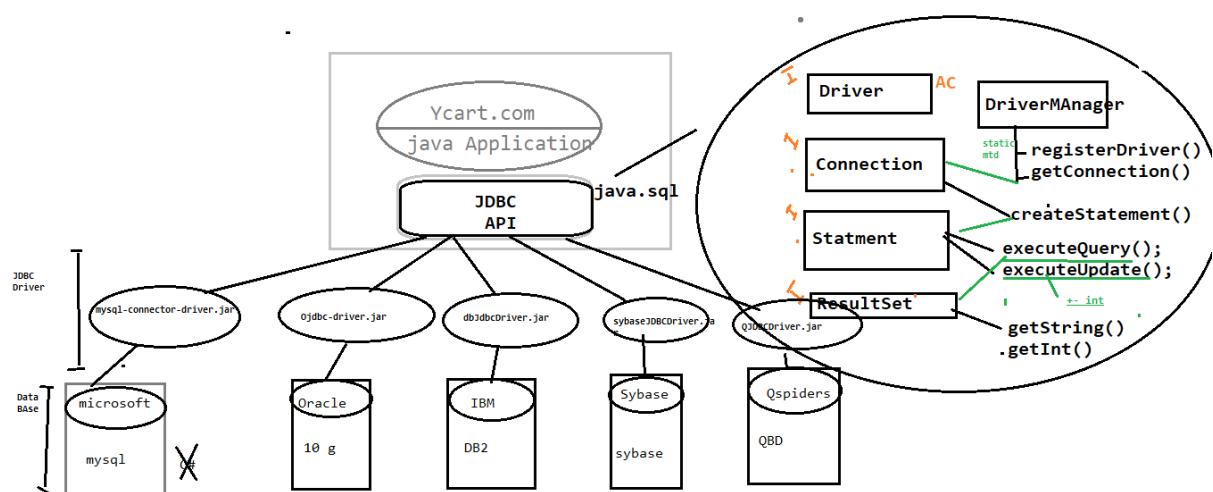
Every Application required physical database to store the data in the form of table

Why JDBC in Development ?

JDBC helps java program to store / retrieve data from the physical database

In Real time, every application required data base to store/retrieve the user information in data base

What is JDBC Drivers & JDBC API ?



JDBC API : its an API's available in `java.sql` package , which is collection interfaces with abstract method & class , & it's database independent

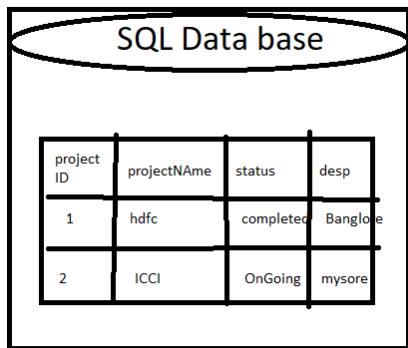
JDBC Driver : its implementation class of JDBC API provided by data base vendor . it's a db dependent

JDBC [Java Data base Connectivity]

There are types of database

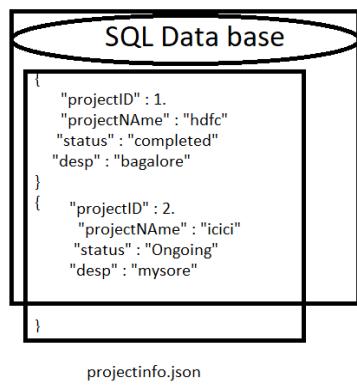
1. SQL Database

- ⇒ Data will be stored in form of table
- EG : mysql , MsSql , Oracle 11 g , DB2 ,

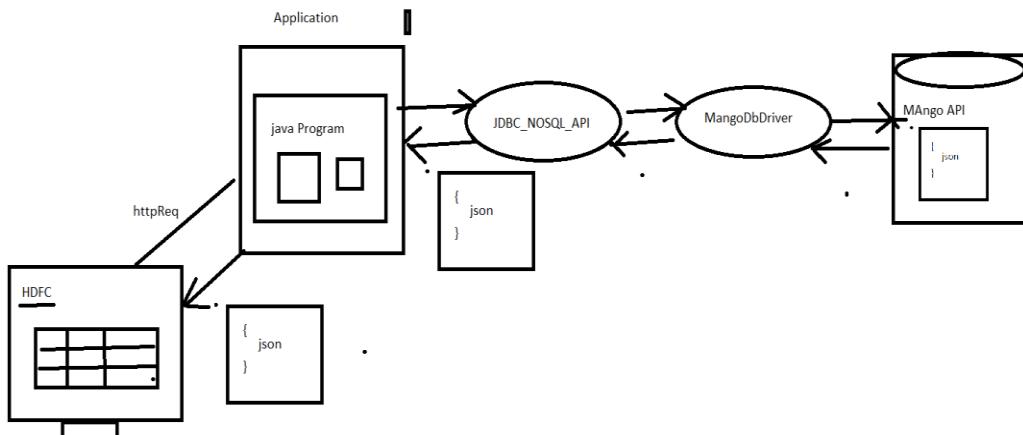


2. No SQL Database

- ⇒ Data will be stored in the from JSON format
- EG : Mango db , Casandra , NoSQL



projectinfo.json



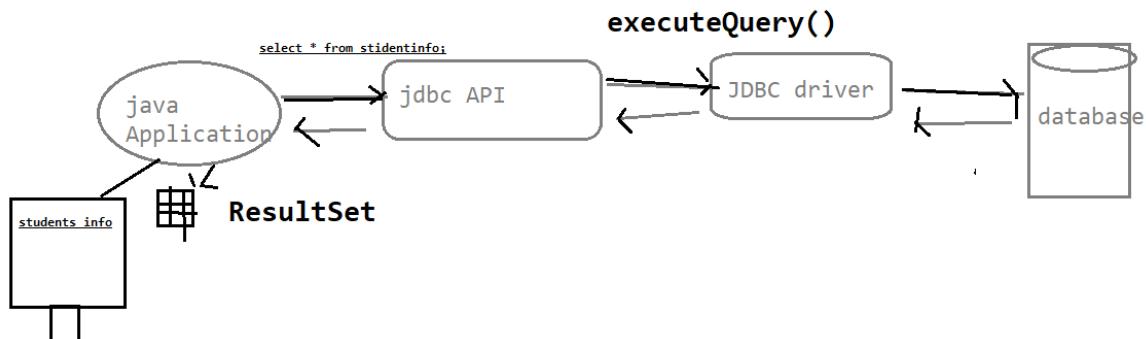
JDBC [Java Data base Connectivity]

Dependency code to get MYSQL drivers

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.27</version>
</dependency>
```

There are two types Query ?

Select Query



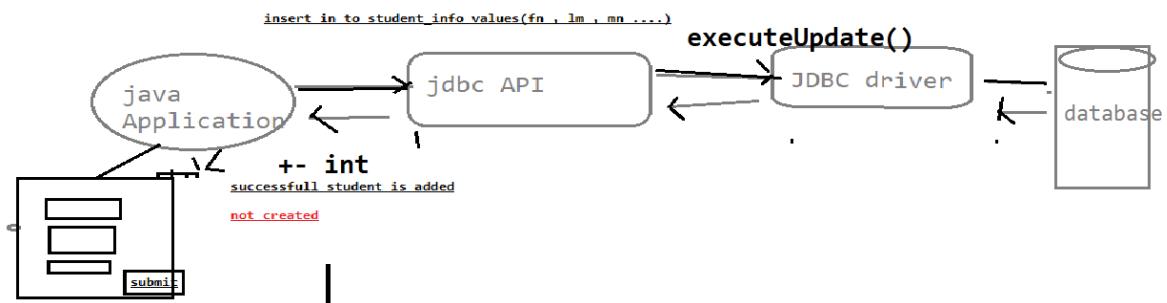
Screenshot of Eclipse IDE showing the code for a Java application named 'SelectQuery_SampleCode.java'. The code demonstrates the JDBC API to execute a SELECT query and handle the resulting ResultSet.

```
package practice;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import com.mysql.jdbc.Driver;
public class SelectQuery_SampleCode {
    public static void main(String[] args) throws Throwable {
        Connection conn = null;
        try {
            Driver driverRef = new Driver();
            /* step 1 : load /register "mysql *the databade" */
            DriverManager.registerDriver(driverRef);

            /* step 2 : connect to db*/
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/projects", "root", "root");
            System.out.println("connection is done");
            /* step 3 : create query statement*/
            Statement stat= conn.createStatement();
            String query = "select * from projec";
            /* step 4 : execute the Query*/
            ResultSet resultSet = stat.executeQuery(query);
            while (resultSet.next()) {
                System.out.println(resultSet.getString(1)+"\t"+resultSet.getString(2)+"\t"
                        +resultSet.getString(3)+"\t"+resultSet.getString(4));
            }
        }catch (Exception e) {
        }finally {
            /*step 5: close the connection */
            conn.close();
            System.out.println("=====close db connection=====");
        }
    }
}
```

JDBC [Java Data base Connectivity]

Non Select Qurey



SDET_29 - Comcast_selenium_Framework/src/test/java/practice/NonSelectQuery_SampleCode2.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
SampleTest.java ReadDataFromProp... commonData.prop... CreateOrgTest.java Comcast_selenium... testdata.xlsx *SelectQuery_Sam... *NonSelectQuery_S... *ProjectUnitTest.j...
11 public class NonSelectQuery_SampleCode2 {
12     public static void main(String[] args) throws Throwable {
13         Connection conn = null;
14         int result = 0;
15         try {
16             Driver driverRef = new Driver();
17             /* step 1 : load /register *mysql *the databade */
18             DriverManager.registerDriver(driverRef);
19             /* step 2 : connect to db*/
20             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/projects", "root", "root");
21             System.out.println("connection is done");
22             /* step 3 : create query statement*/
23             Statement stat = conn.createStatement();
24             String query = "insert into project values('TY_PROJ_3002','deepak','12/01/2022','BankOfBoroda_2','On Going',10)";
25             /* step 4 : execute the Query*/
26             result = stat.executeUpdate(query);
27         }catch (Exception e) {
28     }finally {
29         // TODO: handle exception
30         if(result==1)
31             System.out.println("Project inserted successfully");
32         else {
33             System.err.println("Project is not inserted--!");
34         }
35         /*step 5: close the connection */
36         conn.close();
37         System.out.println("=====close db connection=====");
38     }
39 }
40 }
```

Type here to search Writable Overwrite 39:6:1314 26°C ENG 16:46 12-01-2022

Why JDBC in Automation ?

In Automation also required JDBC Connection , Most of the test case Preconditions , required few set of data in Database & also validation GUI WRT Database

Case 1:

testCase : sendAmount

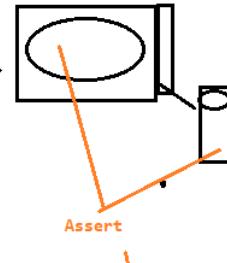
precondition : FAN should have minimum 10 K balance

insert into account values (Accl, active , 10000)

1. login to app
2. navigate to transaction page
3. enter FAN . TAN , Amount(6k) & click on send button
4. verify the sucessfull msg
verify the balance in GUI
verify the balance in Database

select * from account where accountNum = Accl;

testEnv



Case 2

precondition : insert product in to database , which is req for testcase

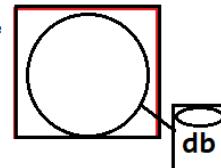
@test

- 1.login to Amazon
2. enter *product* in serch feild
3. click on srach button
4. verify the product list & verify

testData

prodctName	cat	brand	qty
iphone12	apple	pple	10
iphone-7	mobile	apple	20
samsung	mobile	apple	15
moto			

Amazon.com



Case 3

Test Case : crete Order

2 min

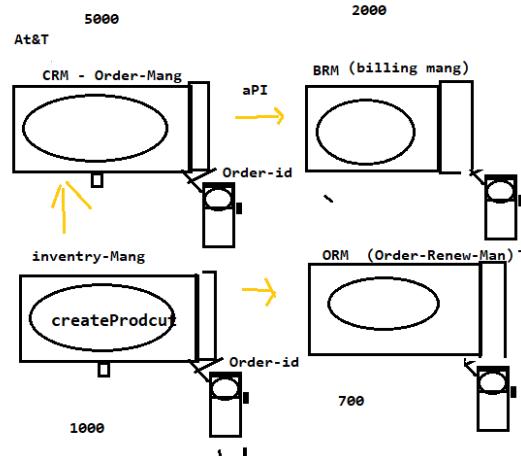
1. login to CRM
2. Go to Odrer page
3. create an Order
4. verify an Order

fk-010

Test Case : BillOnOrder

precondition is | create an order in database via Query

1. login to BRM
2. Go to Billing Page
3. do payemnet
4. verify

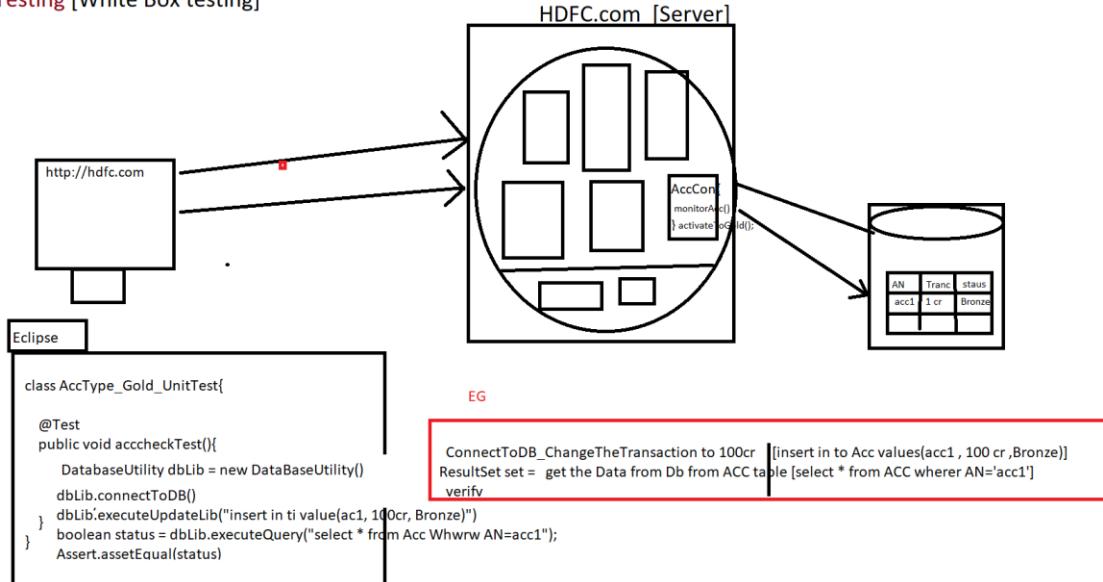


JDBC [Java Data base Connectivity]

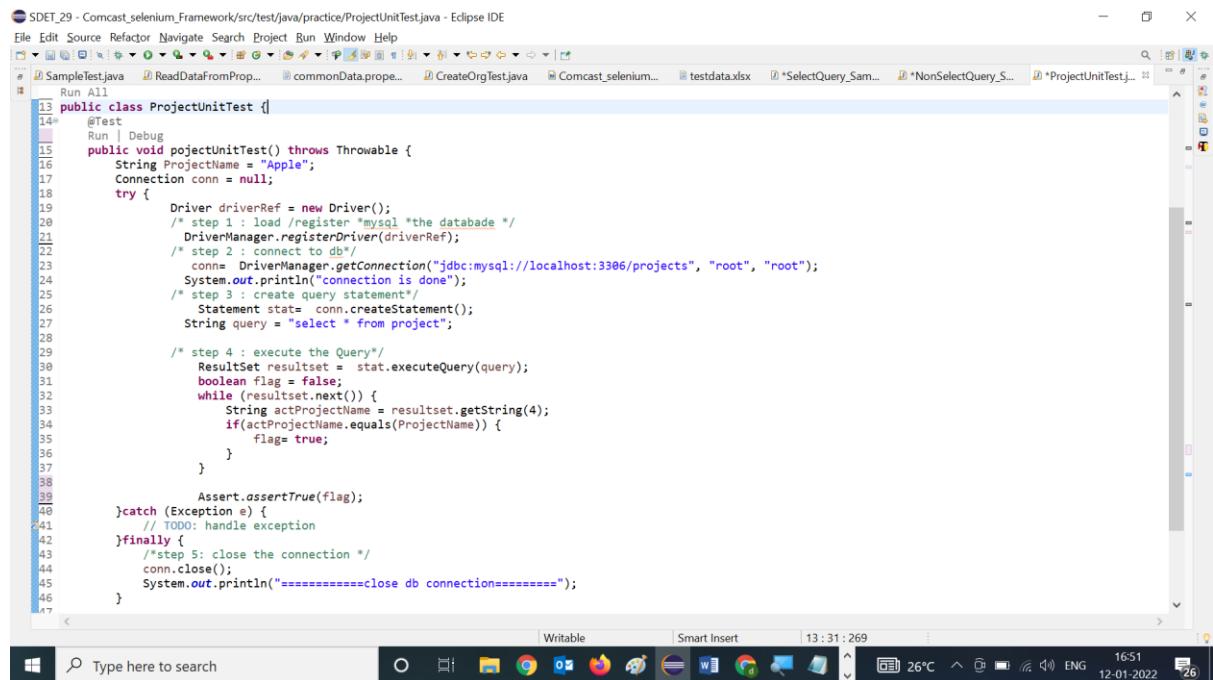
JDBC usage in UNIT Testing[White box Testing]

In case unit testing , [jdbc]database code is required to connect to database & get the data ten verify the data WRT to business logic

Unit Testing [White Box testing]



JDBC [Java Data base Connectivity]



The screenshot shows the Eclipse IDE interface with the title bar "SDET_29 - Comcast_selenium_Framework/src/test/java/practice/ProjectUnitTest.java - Eclipse IDE". The code editor displays Java code for a JDBC test. The code initializes a MySQL driver, connects to a database, creates a statement, executes a query, and asserts the result. The code uses annotations like @Test and imports from java.sql and org.junit.

```
13 public class ProjectUnitTest {
14     @Test
15     public void projectUnitTest() throws Throwable {
16         String ProjectName = "Apple";
17         Connection conn = null;
18         try {
19             Driver driverRef = new Driver();
20             /* step 1 : load /register "mysql *the databade */
21             DriverManager.registerDriver(driverRef);
22             /* step 2 : connect to db*/
23             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/projects", "root", "root");
24             System.out.println("connection is done");
25             /* step 3 : create query statement*/
26             Statement stat= conn.createStatement();
27             String query = "select * from project";
28
29             /* step 4 : execute the Query*/
30             ResultSet resultset = stat.executeQuery(query);
31             boolean flag = false;
32             while (resultset.next()) {
33                 String actProjectName = resultset.getString(4);
34                 if(actProjectName.equals(ProjectName)) {
35                     flag= true;
36                 }
37             }
38             Assert.assertTrue(flag);
39         }catch (Exception e) {
40             // TODO: handle exception
41         }finally {
42             /*step 5: close the connection */
43             conn.close();
44             System.out.println("=====close db connection=====");
45         }
46     }
47 }
```

Generic Libraries

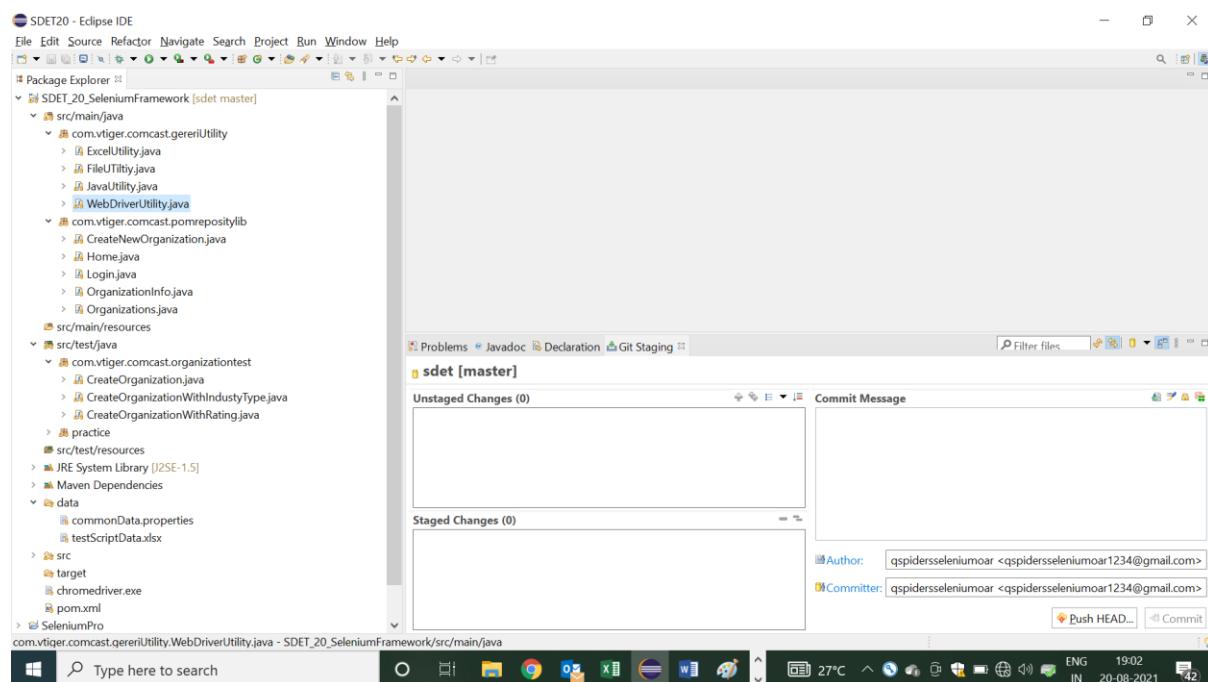
What is Generic components in Automation Framework?

- it's one of the automation framework components which is common for all the application
- its collection of generic class contains reusable methods / libraries
- The methods which can be used to any application is called Generic/common methods

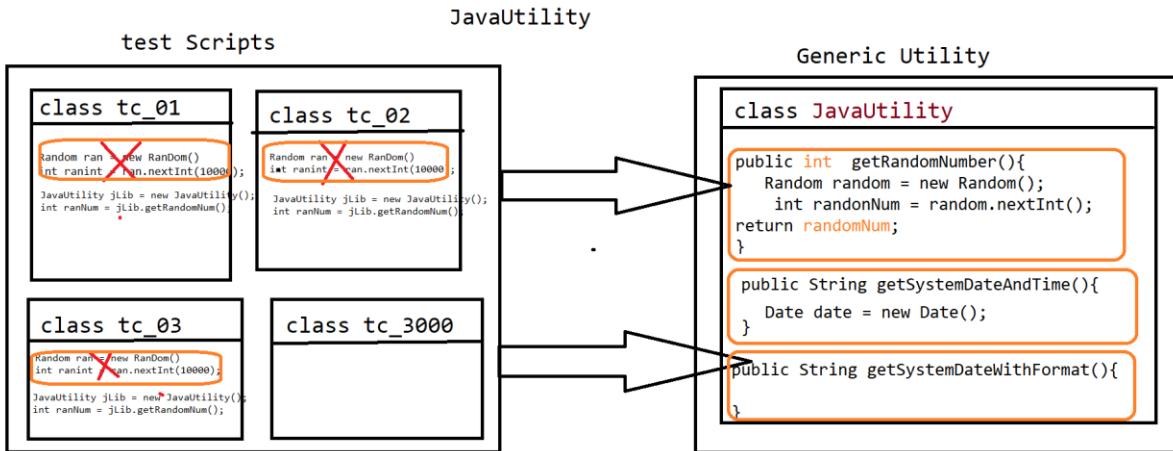
What is the advantages of Generic components?

- ⇒ Reusability of code
- ⇒ Code Optimization
- ⇒ Test script development is faster
- ⇒ Test scripts Code readability
- ⇒ Generic libraries are common to all automation projects
- ⇒ Avoid duplicate Code
- ⇒ no need to remember the syntax every time , just create once & use multiple times

Generic Utility Structure in Automation Project



1. Java Utility Libraries



→ Java Utility is one class in generic component, which contain java specific methods which can be used across the test Scripts / Application

→ its contains several generic reusable methods like

- ⇒ getRandomNum() : it's used to generate random number for every invocation
- ⇒ getSystemDate() : it's used to generate system date and time

=====Code=====

```

package com.vtiger.comcast.genericUtility;

import java.awt.Robot;
import java.awt.event.KeyEvent;
import java.util.Date;
import java.util.Random;

/**
 * this class contains java specific generic libraries
 * @author Deepak
 *
 */
public class JavaUtility {

    /**
     * its used to generate the integer Random number with in the boundary of 0 to 10000
     * @return intData
     */
    public int getRandomNumber() {
        Random ranDom = new Random();
        int ranDomNum = ranDom.nextInt(10000);
        return ranDomNum;
    }

    /**
     * its used to get the current System date & time
     * @return
     */
    public String getSystemDate() {
        Date date = new Date();
        String systemDateAndTime = date.toString();
        return systemDateAndTime;
    }
}

```

```
}

/**
 * its used to get the current System date with YYYY-MM-DD format
 * @return
 */
public String getSystemDate_YYYY_MM_DD() {
    Date date = new Date();
    String systemDateAndTime = date.toString();
    System.out.println(systemDateAndTime);
    String[] arr = systemDateAndTime.split(" ");
    String DD = arr[2];
    String YYYY = arr[5];
    int MM = date.getMonth()+1;

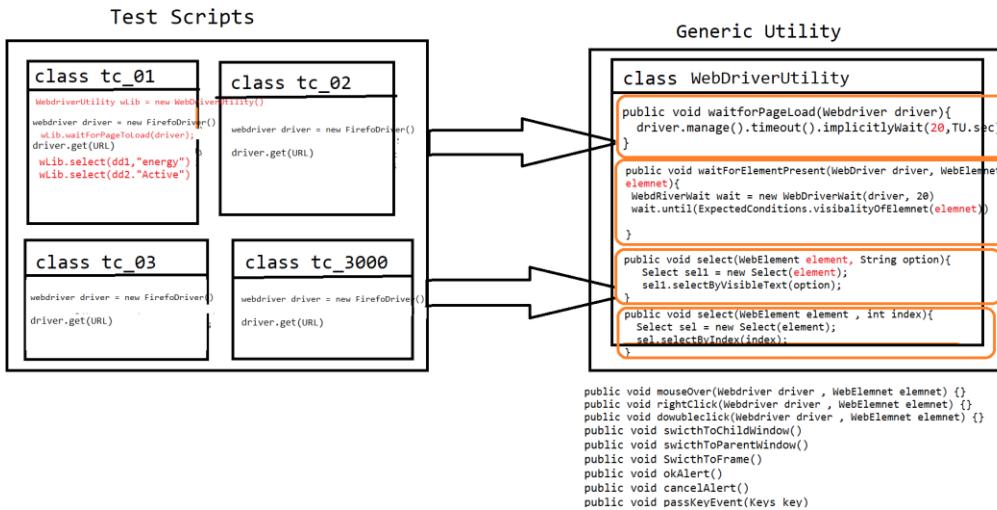
    String finalFormat = YYYY+"-"+MM+"-"+DD;
    return finalFormat;
}

/**
 * used to pass Virtual Key to OS
 * @throws Throwable
 */
public void pressVirtualEnterKey() throws Throwable {

    Robot rc=new Robot();
    rc.keyPress(KeyEvent.VK_ENTER);
    rc.keyRelease(KeyEvent.VK_ENTER);
}

}
```

2. WebDriver Utility Libraries



- ⇒ WebdriverUtility is a Generic class , which contains webdriver specific reusable actions like
- ⇒ waitForPageToLoad()
- ⇒ waitForElement()
- ⇒ select()
- ⇒ acceptAlert()
- ⇒ cancelAlert() .Etc

=====Code=====

```

package com.vtiger.comcast.gererUtility;

import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import java.io.File;
import java.io.IOException;
import java.util.Iterator;
import java.util.Set;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Keys;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.Select;
import org.openqa.selenium.support.ui.WebDriverWait;

import com.google.common.io.Files;

```

```

/**
 * This class contains webdriver specific generic methods
 * Deepak
 */

public class WebDriverUtility {

    /**
     * this method wait for 20 sec for page loading
     * @param driver
     */
    public void waitUntilPageLoad(WebDriver driver)
    {
        driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
    }

    /**
     * This method wait for the element to be visible
     * @param driver
     * @param element
     *
     */
    public void waitForElementVisibility(WebDriver driver, WebElement element)
    {
        WebDriverWait wait = new WebDriverWait(driver, 20);
        wait.until(ExpectedConditions.visibilityOf(element));
    }

    /**
     * This method wait for the element to be clicked , its custom wait created to avoid elemenInterAcceptable Exception
     * @param element
     * @throws throwable
     *
     */
    public void waitAndClick(WebElement element) throws InterruptedException
    {
        int count = 0;
        while(count<20) {
            try {
                element.click();
                break;
            }catch(Throwable e){
                Thread.sleep(1000);
                count++;
            }
        }
    }

    /**
     * this methods enables user to handle dropdown using visible text
     * @param element
     * @param option
     */
    public void select(WebElement element, String option)
    {
        Select select=new Select(element);
        select.selectByVisibleText(option);
    }

    /**
     * this methods enables user to handle dropdown using index
     * @param element
     * @param index
     */
}

```

```

*/
public void select(WebElement element, int index)
{
    Select select=new Select(element);
    select.selectByIndex(index);
}

/**
 * This method will perform mouse over action
 * @param driver
 * @param element
 */
public void mouseOver(WebDriver driver,WebElement element)
{
    Actions act = new Actions(driver);
    act.moveToElement(element).perform();
}

/**
 * This method performs right click operation
 * @param driver
 * @param element
 */
public void rightClick(WebDriver driver,WebElement element)
{
    Actions act = new Actions(driver);
    act.contextClick(element).perform();
}

/**
 * This method helps to switch from one window to another
 * @param driver
 * @param partialWinTitle
 */
public void switchToWindow(WebDriver driver, String partialWinTitle)
{
    Set<String> window = driver.getWindowHandles();
    Iterator<String> it = window.iterator();
    while(it.hasNext())
    {
        String winId=it.next();
        String title=driver.switchTo().window(winId).getTitle();
        if(title.contains(partialWinTitle))
        {
            break;
        }
    }
}

/**
 * Accept alert
 * @param driver
 */
public void acceptAlert(WebDriver driver)
{
    driver.switchTo().alert().accept();
}

/**
 * Cancel Alert
 * @param driver

```

```

        */
    public void cancelAlert(WebDriver driver)
    {
        driver.switchTo().alert().dismiss();
    }
    /**
     * This method used for scrolling action in a webpage
     * @param driver
     * @param element
     */
    public void scrollToWebElement(WebDriver driver, WebElement element) {
        JavascriptExecutor js=(JavascriptExecutor)driver;
        int y= element.getLocation().getY();
        js.executeScript("window.scrollBy(0,"+y+")", element);
    }

    public void switchFrame(WebDriver driver,int index) {
        driver.switchTo().frame(index);
    }

    public void switchFrame(WebDriver driver,WebElement element) {
        driver.switchTo().frame(element);
    }

    public void switchFrame(WebDriver driver,String idOrName) {
        driver.switchTo().frame(idOrName);
    }

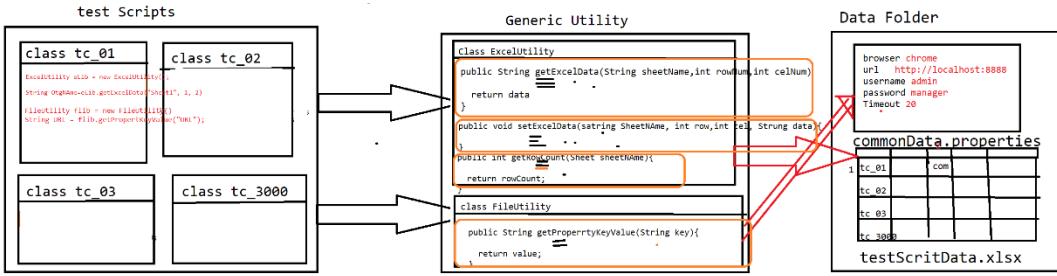
    public void takeScreenshot(WebDriver driver, String screenshotName) throws Throwable {
        TakesScreenshot ts=(TakesScreenshot)driver;
        File src=ts.getScreenshotAs(OutputType.FILE);
        File dest=new File("./screenshot/"+screenshotName+".PNG");
        Files.copy(src, dest);
    }

    /**
     * pass enter Key appertain in to Browser
     * @param driver
     */
    public void passEnterKey(WebDriver driver) {
        Actions act = new Actions(driver);
        act.sendKeys(Keys.ENTER).perform();
    }

}

```

3. Excel Utility libraries



- ⇒ As per the rule of automation, data should not be hardcoded with in the test scripts, so that to get the data from external file like Excel & .properties file We go for ExcelUtility & FileUtility
- ⇒ Excel Utility class is developed using apache Poi libraries, which is used to read the data from Excel
- ⇒ FileUtility is used to get the data from .properties file

=====Code=====

```
package com.vtiger.comcast.gererUtility;

import java.io.FileInputStream;
import java.io.FileOutputStream;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;
/**
 * its developed using Apache POI libraries , which used to handle Microsoft Excel sheet
 * @author Deepak
 */
public class ExcelUtility {
    /**
     * its used read the data from excel base don below arguments
     * @param sheetName
     * @param rowNum
     * @param celNum
     * @return Data
     * @throws Throwable
     */
    public String getDataFromExcel(String sheetName , int rowNum, int celNum) throws Throwable {
        FileInputStream fis = new FileInputStream("./data/testScriptData.xlsx");
        Workbook wb = WorkbookFactory.create(fis);
        Sheet sh = wb.getSheet(sheetName);
        Row row = sh.getRow(rowNum);
        String data = row.getCell(celNum).getStringCellValue();
        wb.close();
        return data;
    }
    /**
     * used to get the last used row number on specified Sheet
     * @param sheetName
     * @return
     * @throws Throwable
     */
    public int getRowCount(String sheetName) throws Throwable {
```

```

        FileInputStream fis = new FileInputStream("./data/testScriptData.xlsx");
        Workbook wb = WorkbookFactory.create(fis);
        Sheet sh = wb.getSheet(sheetName);
        wb.close();
        return sh.getLastRowNum();
    }

    public void setDataExcel(String sheetName , int rowNum, int celNum ,String data) throws
Throwable {
        FileInputStream fis = new FileInputStream("./data/testScriptData.xlsx");
        Workbook wb = WorkbookFactory.create(fis);
        Sheet sh = wb.getSheet(sheetName);
        Row row = sh.getRow(rowNum);
        Cell cel = row.createCell(celNum);
        cel.setCellValue(data);
        FileOutputStream fos = new FileOutputStream("./data/testScriptData.xlsx");
        wb.write(fos);
        wb.close();

    }
}

=====
package com.vtiger.comcast.gereriuility;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

/**
 *
 * @author Deepak
 */
public class FileUTiltiy {
    /**
     * its used to read the data from commonData.properties File based on Key which you pass as an
argument
     * @param key
     * @throws Throwable
     */
    public String getPropertyKeyValue(String key) throws Throwable {
        FileInputStream fis = new FileInputStream("./data/commonData.properties");
        Properties pobj = new Properties();
        pobj.load(fis);
        String value = pobj.getProperty(key);
        return value;
    }
}

```

=====Sample Test Using Generic Utility=====

```

package com.vtiger.comcast.organizationtest;

import java.util.Random;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebDriverException;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;

import com.vtiger.comcast.gereriuility.ExcelUtility;
import com.vtiger.comcast.gereriuility.FileUTiltiy;
import com.vtiger.comcast.gereriuility.JavaUtility;
import com.vtiger.comcast.gereriuility.WebDriverUtility;
import com.vtiger.comcast.pomrepositorylib.CreateNewOrganization;
import com.vtiger.comcast.pomrepositorylib.Home;
import com.vtiger.comcast.pomrepositorylib.Login;
import com.vtiger.comcast.pomrepositorylib.OrganizationInfo;

```

```

import com.vtiger.comcast.pomrepositorylib.Organizations;

public class CreateOrganization {

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

        /*Object Creation for Lib*/
        JavaUtility jLib = new JavaUtility();
        WebDriverUtility wLib = new WebDriverUtility();
        FileUtility fLib = new FileUtility();
        ExcelUtility eLib = new ExcelUtility();

        int randomInt = jLib.getRandomNumber();

        /*common Data*/
        String USERNAME = fLib.getPropertyKeyValue("username");
        String PASSWORD = fLib.getPropertyKeyValue("password");
        String URL = fLib.getPropertyKeyValue("url");
        String BROWSER = fLib.getPropertyKeyValue("browser");

        /*test script Data*/
        String orgName = eLib.getDataFromExcel("Sheet1", 1, 2) + randomInt;

        /* Navigate to app*/
        WebDriver driver = new ChromeDriver();
        wLib.waitUntilPageLoad(driver);
        driver.get(URL);

        /* step 1 : login */
        LoginPage loginPage = new LoginPage(driver);
        loginPage.loginToApp(USERNAME, PASSWORD);

        /*step 2 : navigate to organization*/
        Home homePage = new Home(driver);
        homePage.getOrganizationLnk().click();

        /*step 3 : navigate to "create new organization"page by click on "+" image */
        Organizations orgPage = new Organizations(driver);
        orgPage.getCreateOrgImg().click();

        /*step 4 : create organization*/
        CreateNewOrganization cno = new CreateNewOrganization(driver);
        cno.createOrg(orgName);

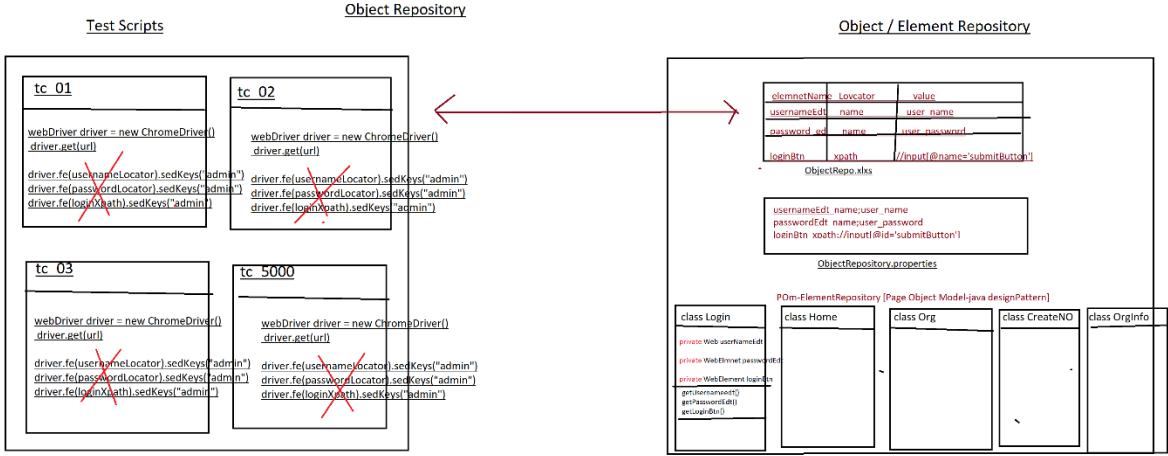
        /*step 5 : verify the successful msg with org name*/
        OrganizationInfo orginfoPage = new OrganizationInfo(driver);
        String actSuccessfullMg = orginfoPage.getSuccessfullMsg().getText();
        if(actSuccessfullMg.contains(orgName)) {
            System.out.println(orgName + "==>created successfully");
        }else {
            System.out.println(orgName + "==> not created successfully");
        }

        /*step 6 : logout*/
        homePage.logout();
    }
}

```

1. What is Object/Elements/POM Repository

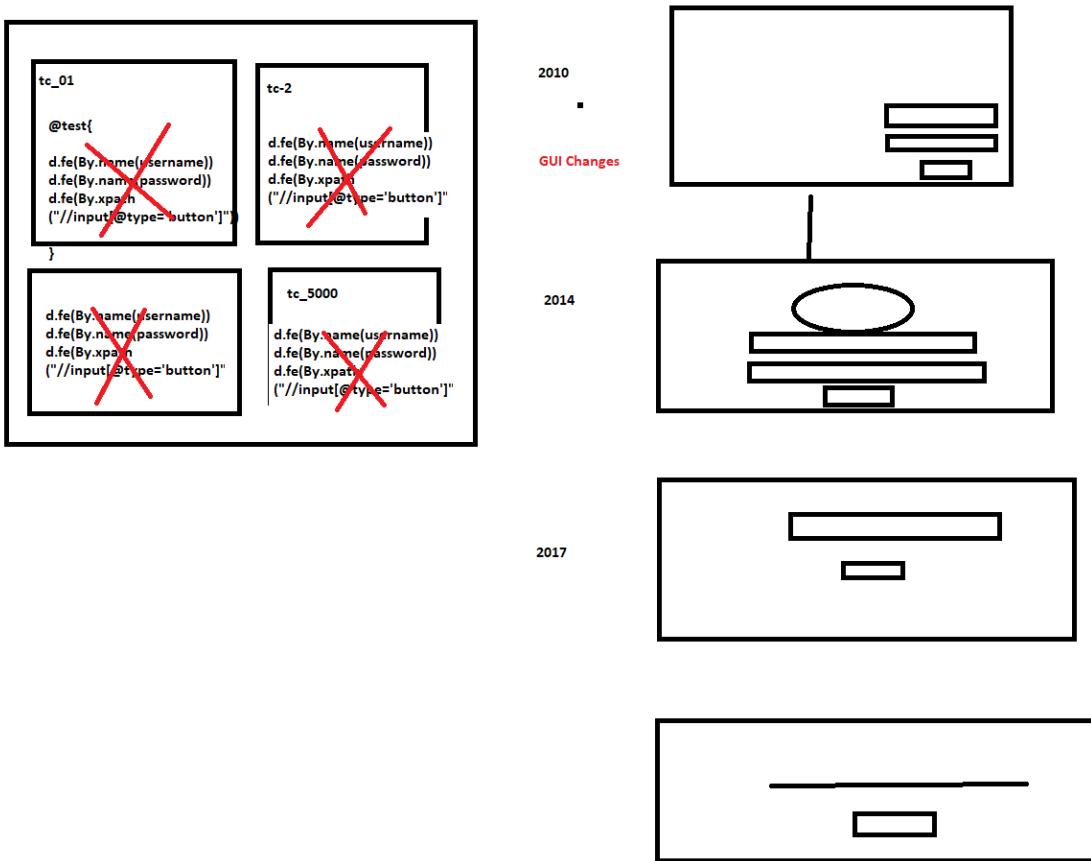
Its collection of elements locators & business libraries in one place & its developed using POM design pattern



2. Why Object repository ?

As per the rule of the automation, we should not hardcode[fixed]elements with in test Scripts instead, we should get elements from Object Repository , because in Agile process due to frequent requirement changes , modification & maintenance of elements is tedious job

EG : below is the example of Gmail application GUI changes



3. What is the advantages repository?

- a. Reusability of elements, no need to write xpath & other locators again & again
- b. Modification in Repository is easy, when GUI changes frequently
- c. Maintenance is easy, because all the elements we kept in one place
- d. Test Script Code Optimized via business reusable libraries
- e. More Readability
- f. Test Script development is faster due to business lib
- g. Test Script is more robust
- h. Handle Stale Elements Exception.

4. What is POM?

POM is a java design pattern preferred by google to develop object repository.

5. Why POM ?

It's a well-organized structured design pattern, where we can maintain all the web elements in page wise, due to POM design pattern maintains & modification is easy & faster.

6. Advantages of POM:

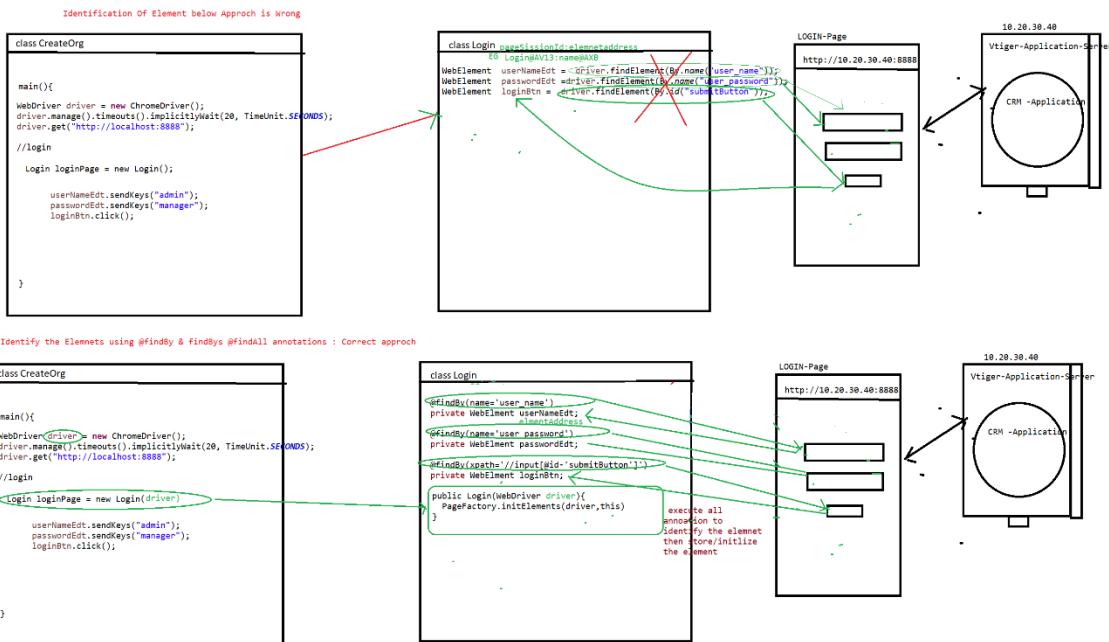
1. Well organized structure
2. Handle stale element exception.
3. maintains & modification of element is easy
4. We can directly store Web Elements in java class
5. Better fit for Agile processes
6. Support Auto heal feature

7. Why @FindBy(locator) annotation instead of driver.findElement("locator")

Ans : to avoid staleElementReferenceException

8. What is staleElementReferenceException ?

It's one of selenium Exception , whenever webdriver try to identify an element , element was available in GUI, but at time of performing an action on the elements element was not recognized due to **page got refreshed** or **elements may become old** or **element not attached to page** in such case we get **staleElementReferenceException**



9. Rules of POM

Rule 1 : create separate java class for every page in a application & class name should be same page name

Rule 2 : Identify all the elements using `@findBy` & `@findAll` , `@findbys` annotations & store them in specific pom / java class (**Element declaration**)

Rule3 : For Every POM class create Constructor to get an Object of the class & initialize the Page Elements , in order to initialize all the page Elements we should use `Pagefactory.initElement()` (**Element initialization**)

Rule 4 : declare all the WebElements as private & provide getters methods to access elements in testScripts class [**this processes is called Encapsulation**]

Note : to create getters mtods inside the java class follow below steps

=>place cursor inside the class → Right click → source → generate getters & setters → select the getters check box → click on ok button

Rule 5 : Go to every & identify the reusable business libraries & implement them in same POM class

10. Difference between POM & PageFactory design pattern?

POM is java design pattern, where will maintain all the Web element locator in well-organized manner

Pagefactory it's an extended design pattern of POM , which is used to create an Object to POM classes , & at the time of object creation it will execute all @findBy @findbys annotation then initialize all the elements

Difference between @findBy , @findAll &@findBys annotation

All annotation available in Selenium webdriver, its traditional ways to identify the elements in GUI.

@findBy : used to identify the element using one locator or one condition

@findAll : it contains multiple @findBy annotation , it mean we can identify the same element using multiple locator (multiple conditions) , it will use OR condition during execution of locator

```
@findALL({ @findBy(@id='username') , @findBy(name='user')})
```

```
Private Webelements userNAmeEdt;
```

Note : using above concepts we can achieve **Autohealing** technique

AutoHealing : during execution , if one locator fails to identify the element , it will retry to identify the same element using another locator

@FindBys : it contains multiple @findBy annotation , it mean we can identify the elements using multiple locator (multiple conditions) , it will use AND condition to during execution of locator

```
@findBys({ @findBy(@id='username') , @findBy(name='user')})
```

```
Private Webelements userNAmeEdt;
```

Project Structure

```
SDET20 - SDET_20_SeleniumFramework/src/test/java/com/vtiger/comcast/organizationtest/CreateOrganization.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
SDET_20_SeleniumFramework
src/main/java
com.vtiger.comcast.pomrepositorylib
CreateNewOrganization.java
Home.java
Login.java
OrganizationInfo.java
Organizations.java
src/main/resources
src/test/java
com.vtiger.comcast.organizationtest
CreateOrganization.java
practice
src/test/resources
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
chromedriver.exe
pom.xml
SeleniumPro
SeleniumProject1
src/main/java
src/main/resources
src/test/java
pac1
image */

43     String actSuccessMsg = orgintopage.getSuccessMsg().getText();
44     if(actSuccessfullMsg.contains(orgName)) {
45         System.out.println(orgName + " ==> created successfully");
com.vtiger.comcast.organizationtest.CreateOrganization.java - SDET_20_SeleniumFramework/src/test/java
Windows Type here to search 26°C ENG IN 17:24 18-08-2021 24
```

Pom Classes : Login

```
SDET20 - SDET_20_SeleniumFramework/src/main/java/com/vtiger/comcast/pomrepositorylib/Login.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrganization.java Home.java *Login.java
1 package com.vtiger.comcast.pomrepositorylib;
2
3 import org.openqa.selenium.WebDriver;
4
5 public class Login { //Rule 1: Create a separate java class for every page in an application
6
7     public Login(WebDriver driver) { // Rule 3: Execute all the elements & initialize the elements PageFactory.initElements [initialization]
8         PageFactory.initElements(driver, this);
9     }
10
11     @FindBy(name = "user_name") //Rule 2: Identify all the elements using @FindBy & findbys , findAll (Declaration )
12     private WebElement userNameEdt;
13
14     @FindBy(name = "user_password")
15     private WebElement userPasswordEdt;
16
17     @FindBy(id="submitButton") , @FindBy(xpath="//input[@id='submitButton']");
18     private WebElement loginBtn;
19
20     //Rule 4: Declare all the elements as private & provide getters method , business method for (Utilization)
21     public WebElement getUserUserNameEdt() {
22         return userNameEdt;
23     }
24
25     public WebElement getUserUserPasswordEdt() {
26         return userPasswordEdt;
27     }
28
29     public WebElement getLoginBtn() {
30         return loginBtn;
31     }
32
33     public void loginToApp(String userName , String password) {
34         /* step 1 : login */
35         userNameEdt.sendKeys(userName);
36         userPasswordEdt.sendKeys(password);
37         loginBtn.click();
38     }
39 }
40
Windows Type here to search Writable Smart Insert 38:9:1421 26°C ENG IN 17:25 18-08-2021 24
```

Pom Classes : Home

SDET20 - SDET_20_SeleniumFramework/src/main/java/com/vtiger/comcast/pomrepositorylib/Home.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrganization.java Home.java Login.java
1 package com.vtiger.comcast.pomrepositorylib;
2
3*import org.openqa.selenium.WebDriver;
4
5 public class Home {
6     WebDriver driver; //global driver variable
7     public Home(WebDriver driver){
8         this.driver = driver;
9         PageFactory.initElements(driver, this);
10    }
11
12    @FindBy(linkText = "Organization")
13    private WebElement organizationLnk;
14
15    @FindBy(linkText = "Products")
16    private WebElement productLnk;
17
18    @FindBy(xpath = "//img[@src='themes/softed/images/user.PNG']")
19    private WebElement administratorImg;
20
21    @FindBy(linkText = "Sign Out")
22    private WebElement singOutLnk;
23
24    public WebElement getOrganizationLnk() {
25        return organizationLnk;
26    }
27
28    public WebElement getProductLnk() {
29        return productLnk;
30    }
31
32    public WebElement getAdministratorImg() {
33        return administratorImg;
34    }
35
36    public WebElement getSingOutLnk() {
37        return singOutLnk;
38    }
39
40    public void logout() {
41        Actions act = new Actions(driver);
42        act.moveToElement(administratorImg).perform();
43        singOutLnk.click();
44    }
45
46
47 }
```

Type here to search Writable Smart Insert 23 : 39 : 694 ENG IN 17:26 18-08-2021 24

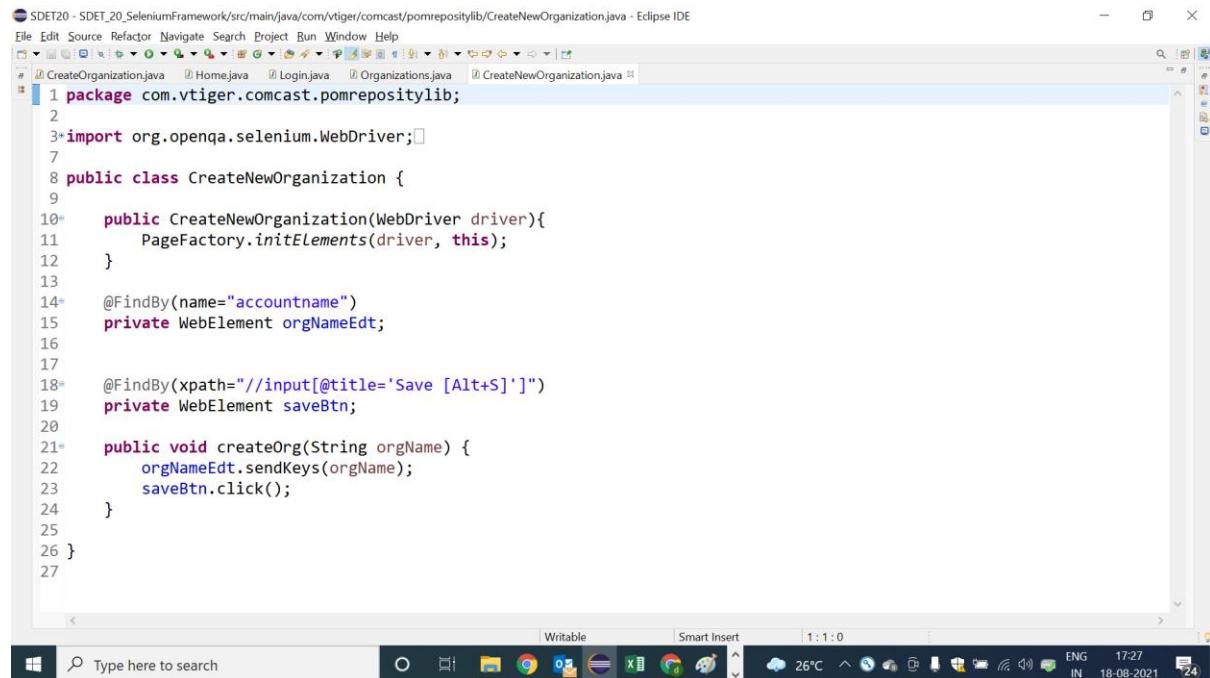
POM class : Organization

SDET20 - SDET_20_SeleniumFramework/src/main/java/com/vtiger/comcast/pomrepositorylib/Organizations.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrganization.java Home.java Login.java Organizations.java
1 package com.vtiger.comcast.pomrepositorylib;
2
3*import org.openqa.selenium.WebDriver;
4
5 public class Organizations {
6
7
8    public Organizations(WebDriver driver) {
9        PageFactory.initElements(driver, this);
10    }
11    @FindBy(xpath = "//img[@src='themes/softed/images/btnL3Add.gif']")
12    private WebElement createOrgImg;
13
14    public WebElement getCreateOrgImg() {
15        return createOrgImg;
16    }
17
18 }
```

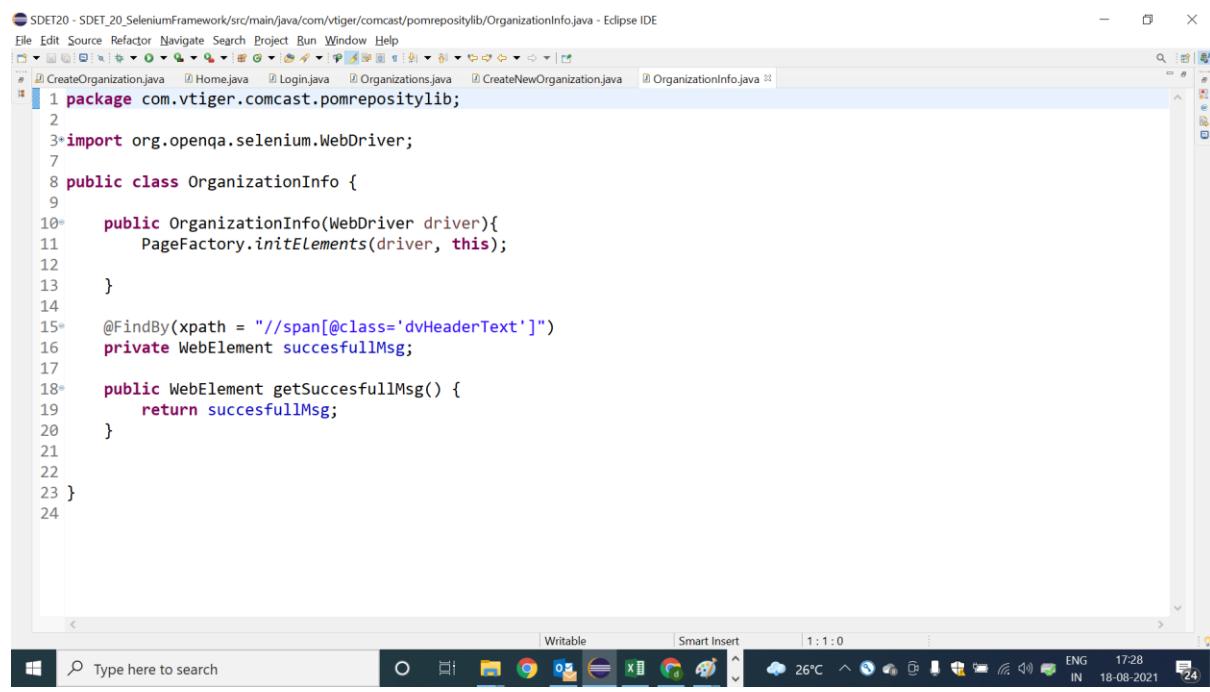
Type here to search Writable Smart Insert 20 : 5 : 520 ENG IN 17:26 18-08-2021 24

POM class : Create new Organization Page



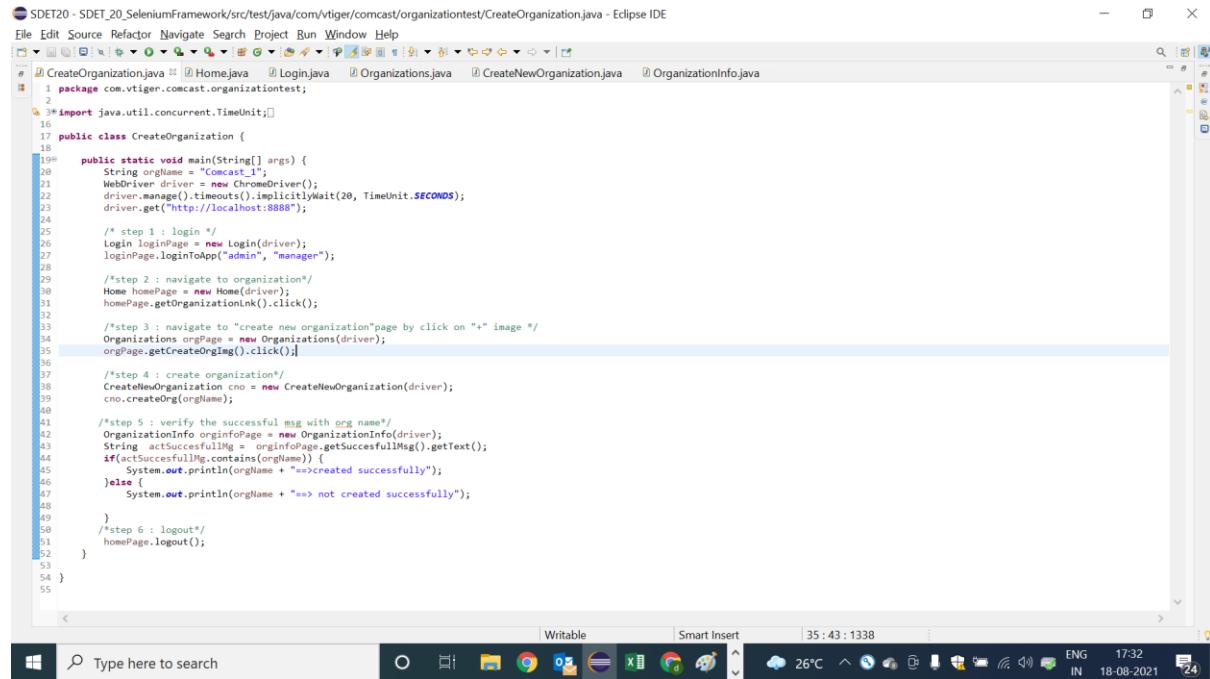
```
SDET20 - SDET_20_SeleniumFramework/src/main/java/com/vtiger/comcast/pomrepositorylib/CreateNewOrganization.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrganization.java Home.java Login.java Organizations.java CreateNewOrganization.java
1 package com.vtiger.comcast.pomrepositorylib;
2
3*import org.openqa.selenium.WebDriver;
4
5 public class CreateNewOrganization {
6
7     public CreateNewOrganization(WebDriver driver){
8         PageFactory.initElements(driver, this);
9     }
10
11     @FindBy(name="accountname")
12     private WebElement orgNameEdt;
13
14     @FindBy(xpath="//input[@title='Save [Alt+S]']")
15     private WebElement saveBtn;
16
17     public void createOrg(String orgName) {
18         orgNameEdt.sendKeys(orgName);
19         saveBtn.click();
20     }
21
22 }
23
24 }
```

POm class : OrganizationInfo



```
SDET20 - SDET_20_SeleniumFramework/src/main/java/com/vtiger/comcast/pomrepositorylib/OrganizationInfo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrganization.java Home.java Login.java Organizations.java CreateNewOrganization.java OrganizationInfo.java
1 package com.vtiger.comcast.pomrepositorylib;
2
3*import org.openqa.selenium.WebDriver;
4
5 public class OrganizationInfo {
6
7     public OrganizationInfo(WebDriver driver){
8         PageFactory.initElements(driver, this);
9     }
10
11     @FindBy(xpath = "//span[@class='dvHeaderText']")
12     private WebElement succesfullMsg;
13
14     public WebElement getSuccesfullMsg() {
15         return succesfullMsg;
16     }
17
18 }
19
20 }
```

Test Scripts using POM class



The screenshot shows the Eclipse IDE interface with a Java file named `CreateOrganization.java` open in the editor. The code implements a POM (Page Object Model) pattern for testing organization creation. It includes imports for `java.util.concurrent.TimeUnit`, `com.vtiger.comcast.organizations.CreateOrganization`, and `com.vtiger.comcast.organizations.CreateNewOrganization`. The main method performs the following steps:

- Step 1: Login to the application.
- Step 2: Navigate to the organization page.
- Step 3: Click on the "Create New Organization" button.
- Step 4: Create a new organization with the name specified in the orgName variable.
- Step 5: Verify that the success message contains the organization name.
- Step 6: Logout from the application.

```
SDET20 - SDET_20_SeleniumFramework/src/test/java/com/vtiger/comcast/organizationtest/CreateOrganization.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrganization.java Home.java Login.java Organizations.java CreateNewOrganization.java OrganizationInfo.java
1 package com.vtiger.comcast.organizations;
2 import java.util.concurrent.TimeUnit;
3 public class CreateOrganization {
4     public static void main(String[] args) {
5         String orgName = "Comcast 1";
6         WebDriver driver = new ChromeDriver();
7         driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
8         driver.get("http://localhost:8888");
9
10        /* step 1 : login */
11        LoginPage loginPage = new LoginPage(driver);
12        loginPage.loginToApp("admin", "manager");
13
14        /*step 2 : navigate to organization*/
15        HomePage homePage = new HomePage(driver);
16        homePage.getOrganizationLnk().click();
17
18        /*step 3 : navigate to "create new organization"page by click on "+" image */
19        Organizations orgPage = new Organizations(driver);
20        orgPage.getCreateOrgImg().click();
21
22        /*step 4 : create organization*/
23        CreateNewOrganization cno = new CreateNewOrganization(driver);
24        cno.createOrg(orgName);
25
26        /*step 5 : verify the successful msg with org name*/
27        OrganizationInfo orgInfoPage = new OrganizationInfo(driver);
28        String actSuccessfulMsg = orgInfoPage.getSuccessfulMsg().getText();
29        if(actSuccessfulMsg.contains(orgName)) {
30            System.out.println(orgName + "=>created successfully");
31        } else {
32            System.out.println(orgName + "=> not created successfully");
33        }
34        /*step 6 : logout*/
35        homePage.logout();
36    }
37 }
38 }
```

```
SDET20 - SDET_20_SeleniumFramework/src/test/java/com/vtiger/comcast/organizationtest/CreateOrganization.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrganization.java Home.java Login.java Organizations.java CreateNewOrganization.java OrganizationInfo.java
1 package com.vtiger.comcast.organizationtest;
2
3*import java.util.concurrent.TimeUnit;
4
5 public class CreateOrganization {
6
7     public static void main(String[] args) {
8         String orgName = "Comcast_1";
9         WebDriver driver = new ChromeDriver();
10        driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
11        driver.get("http://localhost:8888");
12
13        /* step 1 : login */
14        LoginPage loginPage = new LoginPage(driver);
15        loginPage.loginToApp("admin", "manager");
16
17        /*step 2 : navigate to organization*/
18        HomePage homePage = new HomePage(driver);
19        homePage.getOrganizationLnk().click();
20
21        /*step 3 : navigate to "create new organization"page by click on "+" image */
22        Organizations orgPage = new Organizations(driver);
23        orgPage.getCreateOrgImg().click();
24
25        /*step 4 : create organization*/
26        CreateNewOrganization cno = new CreateNewOrganization(driver);
27        cno.createOrg(orgName);
28
29        /*step 5 : verify the successful msg with org name*/
30        OrganizationInfo orginfoPage = new OrganizationInfo(driver);
31        String actSuccessfullMsg = orginfoPage.getSuccessfullMsg().getText();
32        if(actSuccessfullMsg.contains(orgName)) {
33            System.out.println(orgName + "=>created successfully");
34        }else {
35            System.out.println(orgName + "=> not created successfully");
36        }
37
38        /*step 6 : logout*/
39        homePage.logout();
40    }
41
42 }
43
44 }
```

```
SDET20 - SDET_20_SeleniumFramework/src/test/java/com/vtiger/comcast/organizationtest/CreateOrganization.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrganization.java Home.java Login.java Organizations.java CreateNewOrganization.java OrganizationInfo.java
30        homePage.getOrganizationLnk().click();
31
32        /*step 3 : navigate to "create new organization"page by click on "+" image */
33        Organizations orgPage = new Organizations(driver);
34        orgPage.getCreateOrgImg().click();
35
36        /*step 4 : create organization*/
37        CreateNewOrganization cno = new CreateNewOrganization(driver);
38        cno.createOrg(orgName);
39
40        /*step 5 : verify the successful msg with org name*/
41        OrganizationInfo orginfoPage = new OrganizationInfo(driver);
42        String actSuccessfullMsg = orginfoPage.getSuccessfullMsg().getText();
43        if(actSuccessfullMsg.contains(orgName)) {
44            System.out.println(orgName + "=>created successfully");
45        }else {
46            System.out.println(orgName + "=> not created successfully");
47        }
48
49        /*step 6 : logout*/
50        homePage.logout();
51    }
52
53
54 }
```

Unit Testing Framework Tool

Available Unit testing framework tools in Current Market :

TestNG -----> Java, .Net	[Eclipse / JDevelopers / IntelliJ]
Junit -----> Java	[Eclipse / JDevelopers / IntelliJ]
Nunit -----> .Net	[Visual Studio]
Pydev -----> Python	[PyChram]
Rspsc -----> Ruby	[Eclipse]
Jasmin ➔ javascript	[WebStrom]

All unit testing framework tool is implemented as plugin for eclipse IDE, but Junit is a default plugin for eclipse IDE.

What is testNG ?

- ➔ TestNG is a unit test **TDD**[test Driven Development] framework , which support java & .Net
- ➔ TestNG is an open source unit test framework tool, where NG stands for **Next Generation**.
- ➔ TestNg developed as addition plugin for Eclipse
- ➔ TestNG is inspired from JUNIT & NUNIT, it means it has all the features of Nunit & Junit & also contains additional features that make TestNg become more powerFull

Installation steps of TestNG:

Go to Eclipse window

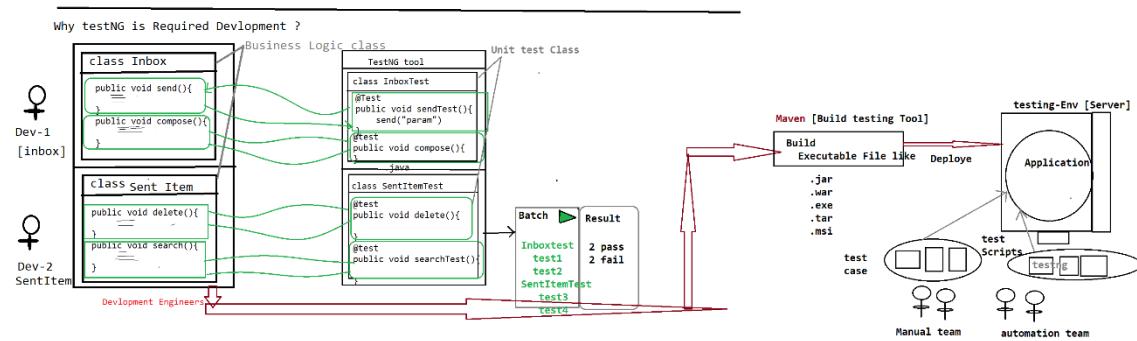
Click on help option-> Eclipse Marketplace

- ➔ Write TestNG in find edit box and click on go button.

- ➔ Find TestNG for eclipse division and click on **install** button
- ➔ Click on confirm button and I accept the terms and conditions and click on finish
- ⇒ **In order to verify the TestNG installation** → Go to windows
- Show view → others → expand java folder, TestNG symbol will be present

TestNG usage in development:

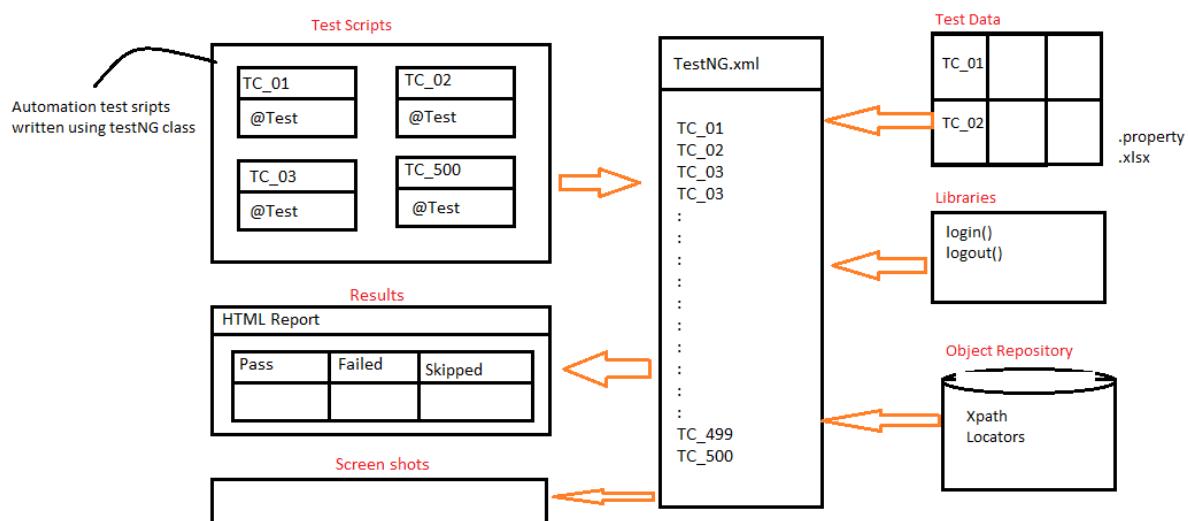
TestNG is used in development to write white box/ unit test cases and each unit test case will be used to test the source code of the application



➔ In case of development, testNG will be used to develop unit test cases and each unit test case check the business logic of the source code.

➔ Used to achieve WBT

TestNG usage in selenium automation



➔ In case of automation, testNG will be used to develop all the scripts using testNG

annotations and achieve batch execution without any manual interaction.

→ TestNG will be used to handle all framework component & help us to run all the test scripts in batch / parallel / group without any manual intervention

→ TestNG.xml is main controller of the selenium framework , where we start the execution

Why TestNG ? Why not Junit ?

- Annotation
- Batch Execution
- Assesrtions

New functionalities are:

- >Html report
- >Parallel execution
- >Grouping execution
- >Additional annotations
- >batch execution is easier
- >iTest Listeners [used to take ScrrenShot]
- >Retry Analyser [used to rerun the failed test script]

Annotations:

There are 4 blocks in java

1. classs Block
2. Interface Block
3. Enum block
4. Annoation block

class A{
}

interface I{
}

Enum Key{
}

Annoation Test{
}

Its Java block , which is used to provide metadata(information/instruction) to the JVM , at the time of execution in RUN-Time.

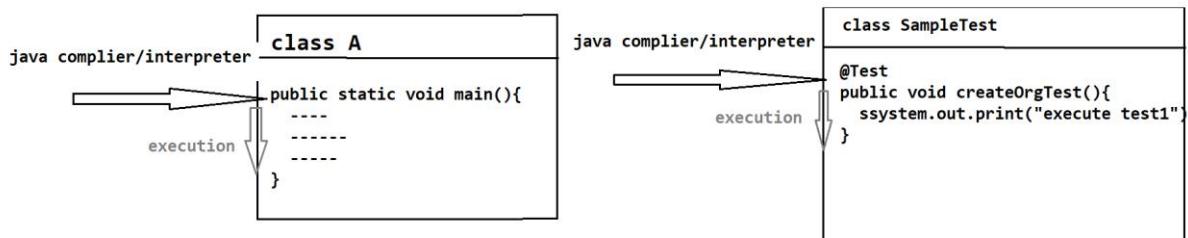
⇒ Annotation always start with @symbol

- @Test
- @BeforeMethod
- @AfterMethod
- @BeforeClass
- @AfterClass

=====addition annotation available only in TestNG

- @BeforeTest
- @AfterTest
- @beforeSuite
- @afterSuite
- @paramaters
- @dataProvider
- @Listner

@Test



The screenshot shows the Eclipse IDE interface with the title bar "workspace_SDET - SeleniumProject/src/pac/CustomerTest.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Copy, Paste, Find, and Run. The left sidebar shows the project structure with "CustomerTest.java" selected. The main editor area displays the following Java code:

```
1 package pac;
2
3 import org.testng.annotations.Test;
4
5 public class CustomerTest {
6
7     @Test
8     public void createCustomerTest() {
9         System.out.println("execute HDFC createCustomerTest");
10    }
11
12     @Test
13     public void modifyCustomerTest() {
14         System.out.println("execute HDFC modifyCustomerTest");
15    }
16
17     @Test
18     public void deleteCustomerTest() {
19         System.out.println("execute HDFC deleteCustomerTest");
20    }
21
22
23
24 }
```

The status bar at the bottom shows "Writable", "Smart Insert", "22:1", and the system tray indicates the date and time as "12:46 PM 8/10/2019".

1. Whenever we execute testng class, javaCompiler/Interpreter always looks for @Test Annotation method to start the execution.
 2. **Without @Test , testNG class will not be executed, @test annotation method act like main method in testNG**
 3. In one testng class we can have multiple @test methods, but each test method should have @Test annotation before method signature.
 4. Annotation method return type should be “void” and access specifier should be public., but method name can be anything (we have provide manual testNameTest).
 5. As per the Rule of the Automation , TESTNG class Name should be ModuleNAmE , @test method name should be manual testCase Name
 6. As per the Rule TestNG class Name & testNG method Name should end With “Test”
 7. One Manual test case contains multiple steps all those steps should be automated using one @test annotation & test name should be manual test-case Name & end with Test
 8. In one class we can keep multiple @test annotation , but in real time we are going maintain maximum 10 to 15 @test script , because maintenance will be easy

How to Verify html report

Refresh the project after execution—select project right click and click on refresh

Automatically we get **test-Output** folder within the same project.

Expand test-output folder → select emailable.html and right click → open with → Open with browser

Priority

Whenever we execute testNG class , by default all the test method will be executed based on Alphabetical Order , in order to change the Order of Execution , we go for priority

The screenshot shows the Eclipse IDE interface with the following details:

- Customer.java:** The code defines a class Customer with three methods: createCustomerTest, modifyCustomerTest, and deleteCustomerTest. Each method is annotated with @Test and has a priority attribute (1, 2, and 3 respectively). The code also includes System.out.println statements to log the execution of each method.
- Console Output:** The output shows the execution results:

```
<terminated> Customer [TestNG] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (Aug 10, 2019, 12:19:22 PM)
PASSED: createCustomerTest
PASSED: modifyCustomerTest
PASSED: deleteCustomerTest

=====
Default test
Tests run: 3, Failures: 0, Skips: 0
=====
```
- Bottom Status Bar:** Shows the system tray icons and the date/time (12:20 PM, 8/10/2019).

DependsOnMethod :

It's help us to check the dependent test case is pass or fail,

If dependent test-script get pass, execution will continue

If dependent test-script get fail, skip the all other test script which is dependent on first test

The screenshot shows the Eclipse IDE interface. The top window displays the code for `Customer.java`, which contains several test methods using the `@Test` annotation. The bottom window is the TestNG console, showing the execution results for the `Customer` class. The console output indicates a total of 3 tests run, 1 failure, and 2 skips. The time taken for the execution is 29 ms.

```
workspace_SDET - SeleniumProject/src/pac/Customer.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Customer.java
4
5 public class Customer {
6
7     @Test
8     public void createCustomerTest() {
9         System.out.println("execute HDFC createCustomerTest");
10        int[] arr = {1,2,3};
11        System.out.println(arr[5]);
12    }
13    @Test(dependsOnMethods="createCustomerTest")
14    public void modifyCustomerTest() {
15        System.out.println("execute modify HDFC to AIRTel CustomerTest");
16    }
17    @Test(dependsOnMethods="modifyCustomerTest")
18    public void deleteCustomerTest() {
19        System.out.println("execute delete AIRTEL CustomerTest");
20    }
}

Console
<terminated> Customer [TestNG] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (Aug 10, 2019, 12:29:03 PM)
=====
Default suite
Total tests run: 3, Failures: 1, Skips: 2
=====

[TestNG] Time taken by org.testng.reporters.jq.Main@140d5f0: 29 ms

12:30 PM
8/10/2019
```

10. Invocation Count:

Same test-script executed with multiple Times with same test data

The screenshot shows the Eclipse IDE interface. The top window displays the code for `BankSendAmountTest.java`, which includes an annotation `@Test(invocationCount=4)` on the `sendAmount` method. The bottom window is the TestNG console, showing the execution results. The console output indicates 4 passes for the `sendAmount` test, resulting in a total of 4 tests run, 0 failures, and 0 skips.

```
workspace_SDET - SeleniumProject/src/pac/BankSendAmountTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
BankSendAmountTest.java
4
5 public class BankSendAmountTest {
6
7     @Test(invocationCount=4)
8     public void sendAmount() {
9         System.out.println("execute send Amount 50K");
10    }
11 }
12
13

Console
<terminated> BankSendAmountTest [TestNG] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (Aug 10, 2019, 12:35:56 PM)
PASSED: sendAmount
PASSED: sendAmount
PASSED: sendAmount
PASSED: sendAmount
=====

Default test
Tests run: 4, Failures: 0, Skips: 0
=====

=====
Default suite
1 Writable | Smart Insert | 7 : 26
12:38 PM
8/10/2019
```

11 . Data Provider

- a. In Order to execute same test case multiple Times with different test Data , we go for @DataProvider annotation
- b. Data Provider annotation always return TWO -DIMENSTINAL Object array , because we can pass any type of datatype
- c. Data Provider annotation help us to execute same test multiple times with the different set of data , each test-script should have dedicated @dataProvider annotation
- d. DataProvider annotation play major role in Data driven framework , where we need to test the application with huge amount of data like Ecommerce , Booking ,banking application
- e. In Below example , row count is 5 ➔ it indicates test needs to be executed 5 times

Column count is 2 ➔ it indicates every iteration 2 arguments will be passed

object[5][2]

Banglaore ^{0,0}	Mysore ^{0,1}
Banglore ^{1,0}	Goa ^{1,1}
Banglore ^{2,0}	Mangalore ^{2,1}
^{3,0}	^{3,1}
Banglore	Kerala
Banglore ^{4,0}	Mumbai ^{4,1}

→ Sample program for data provider

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** SDT24 - SDT_24_MavenProject/src/test/java/practice/BookTicket.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar items.
- Package Explorer:** Shows the project structure with packages like db_practice, practice, and data, along with various Java files and resources.
- Code Editor:** Displays the following Java code:

```
5     Run All
6     public class BookTicket {
7         @Test(dataProvider = "dataProvider_bookTicketTest" )
8         Run | Debug
9         public void bookTicketTest(String src , String dst) {
10             System.out.println("Book titcket from "+src+" to "+dst);
11         }
12         @DataProvider
13         public Object[][] dataProvider_bookTicketTest() {
14             Object[][] objArr = new Object[5][2];
15
16             objArr[0][0] = "Banglore";
17             objArr[0][1] = "Goa";
18
19             objArr[1][0] = "Banglore";
20             objArr[1][1] = "mysore";
21
22             objArr[2][0] = "Banglore";
23             objArr[2][1] = "Managalore";
24
25             objArr[3][0] = "Banglore";
26             objArr[3][1] = "Hyd";
27
28             objArr[4][0] = "Banglore";
29             objArr[4][1] = "MP";
30             return objArr;
31         }
32     }
```

- Bottom Status Bar:** Shows the date and time (02-11-2021, 18:01), system status (ENG IN), and battery level.

→ Data provider example with 3 arguments

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** SDT20 - SDT_20_SeleniumFramework/src/test/java/practice/SampleTestDataProvider.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar items.
- Package Explorer:** Shows the project structure with packages like SDT_20_Selenium, practice, and SampleTest, along with various Java files and resources.
- Code Editor:** Displays the following Java code:

```
1 package practice;
2
3 import org.testng.annotations.DataProvider;
4
5 Run All
6 public class SampleTestDataProvider {
7
8     @Test(dataProvider = "bookTicketDataProvider")
9     Run | Debug
10    public void bookTicket(String src , String dest , int ticket) {
11        System.out.println("execute source =>" +src + ", Destination=>" +dest);
12    }
13
14    @DataProvider
15    public Object[][] bookTicketDataProvider() {
16        Object[][] objArr = new Object[5][3];
17        objArr[0][0] = "Banglore";
18        objArr[0][1] = "Mysore";
19        objArr[0][2] = 10;
20
21        objArr[1][0] = "Banglore";
22        objArr[1][1] = "Goa";
23        objArr[1][2] = 10;
24
25        objArr[2][0] = "Banglore";
26        objArr[2][1] = "Magalore";
27        objArr[2][2] = 10;
28
29        objArr[3][0] = "Banglore";
30        objArr[3][1] = "Kerala";
31        objArr[3][2] = 10;
32
33        objArr[4][0] = "Banglore";
34        objArr[4][1] = "Mumbai";
35        objArr[4][2] = 10;
36        return objArr;
37    }
38}
```

- Output View:** Shows the execution results of the test cases.
- Bottom Status Bar:** Shows the date and time (23-08-2021, 15:27), system status (ENG 23-08-2021), and battery level.

@BeforeMethod @AfterMethod

1. Before method annotations will be executed, before executing each @test method in a class
2. After method annotation will be executed, after executing each @test in a class
3. Beforemethod & aftermethod will not be executed, without @test annotation method, because they are configuration method
4. In order to implement similar pre-condition for all the testcase like “LOGIN code” we go for @BM
5. In order to implement similar post-condition for all the testcase like “LOGOUT code” we go for @AM

The screenshot shows the Eclipse IDE interface with the following details:

- Project Structure:** The Package Explorer view shows the project structure for "SDET_20_SeleniumFramework". It includes packages for main/java, test/java, and test/resources, along with JRE System Library (J2SE-1.5), Maven Dependencies, TestNG, JUnit 5, and SeleniumPro.
- Code Editor:** The editor displays the Java code for `CreateOrgTest.java`. The code includes imports for `org.testng.annotations.BeforeMethod` and `org.testng.annotations.AfterMethod`. It defines a class `CreateOrgTest` with methods `configBM()`, `configAM()`, `createOrgTest()`, `createOrgWithIndustriesTest()`, and `createOrgWithRatingTest()`. Each method contains logic to print to `System.out` indicating steps like "Launch the Browser", "login", "logout", and "verify".
- Output View:** The right-hand pane shows the TestNG execution results. It lists the tests run: `createOrgTest`, `createOrgWithIndustriesTest`, and `createOrgWithRatingTest`. The output indicates that all three tests passed. The log also shows the sequence of operations: launching the browser, logging in, creating an organization with industries, verifying, logging out, and closing the browser.
- Taskbar:** At the bottom, the Windows taskbar shows the system tray with icons for battery, network, and date/time (23-08-2021, 12:53).

@BeforeClass & @Afterclass

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "SDTE_20_SeleniumFramework".
- CreateOrgTest.java Content:** A Java test class with methods for creating organizations and navigating between modules.
- Terminal Output:** The right-hand terminal window displays the execution results of the test cases, showing three passed tests and one failed test.

```
SDTE20 - SDTE_20_SeleniumFramework/src/test/java/practice/CreateOrgTest.java - Eclipse IDE
File Edit Source Refactor Navigate Project Run Window Help
SDTE_20_SeleniumFramework [sdte master]
src/main/java
src/main/resources
src/test/java
com.vtiger.comcast.organizationtest
practice
CreateOrgTest_BC_AC.java
CreateOrgTest.java
OrgTest.java
Sample_Date_test.java
Sample_Random.java
SampleCreateOrganization.java
SampleTest_dataProvider.java
src/test/resources
JRE System Library [J2SE-1.5]
Maven Dependencies
TestNG
JUnit 5
data
src
target
output
chromedriver.exe
pom.xml
SeleniumPro
SeleniumProject

CreateOrgTest.java
import org.testng.annotations.Test;
Run All
public class CreateOrgTest {
@BeforeClass
public void configBC() {
System.out.println("*****launch Browser*****");
}
@AfterMethod
public void configWM() {
System.out.println("==login==");
}
@Test
public void createOrgTest() {
System.out.println("navigate to Org Module");
System.out.println("create Org");
System.out.println("verify");
}
@Test
public void createOrgWithIndustriesTest() {
System.out.println("navigate to Org Module");
System.out.println("create Org with industries");
System.out.println("verify");
}
@Test
public void createOrgWithRatingTest() {
System.out.println("navigate to Org Module");
System.out.println("create Org with rating");
System.out.println("verify");
}
@AfterMethod
public void configWM() {
System.out.println("==logout==");
}
@AfterClass
public void configAC() {
System.out.println("*****lose Browser*****");
}
}
47

<terminated> CreateOrgTest [TestNG] detected TestNG version 7.3.0
=====
log4j:WARN No appenders could be found for logger (org.openqa.selenium.htmlunit.HtmlUnitDriver).
log4j:WARN Please initialize the log4j system properly.
=====
===== navigate to Org Module
===== create Org
===== verify
===== ==logout==
===== ==login==
===== navigate to Org Module
===== create Org with industries
===== create Org with rating
===== verify
===== ==logout==
===== ==login==
===== navigate to Org Module
===== create Org with rating
===== verify
===== ==logout==
===== =====lose Browser=====
PASSED: createOrgTest
PASSED: createOrgWithIndustriesTest
PASSED: createOrgWithRatingTest
=====
Default test
Tests run: 3, Failures: 0, Skips: 0
=====
=====
Default suite
Total tests run: 3, Passes: 3, Failures: 0, Skips: 0
23-08-2021 13:07
```

1. BeforeClass Annotation method will be executed, before Executing first @test in a class
 2. AfterClass Annotation method will be executed, after executing all/last test-case with in a class
 3. BeforeClass & AfterClass annotations will be executed only once in a entire class execution.
 4. It will be used to develop global configuration like launch browser, object initialization

NOTE: As per the Rule of the Automation , test case should not have dependency between one to another test , every testcase should be unique (it means every test should have fresh login & logout code)

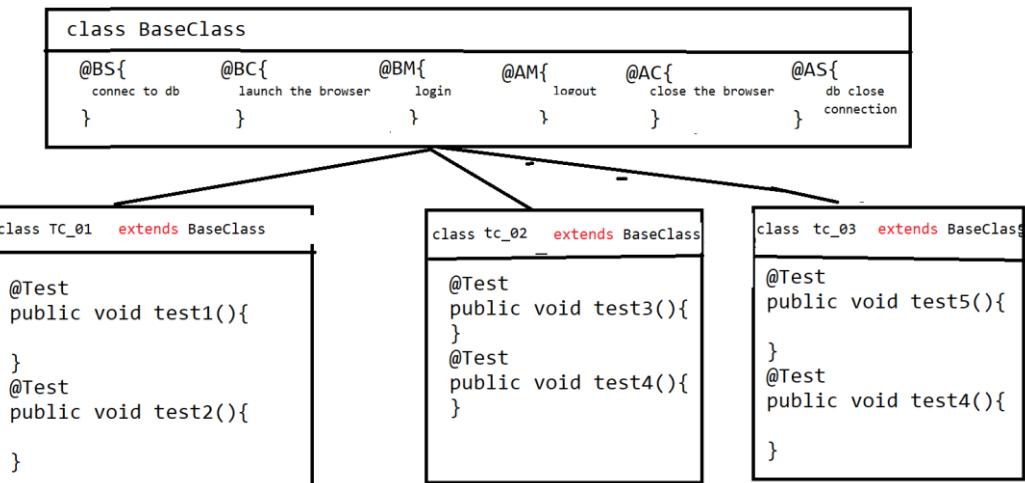
ConFig Annotation methods Usage in REAL Selenium Framework

1. In Real selenium FrameWork , all the configure annotation will be implemented Inside the BaseClass, that is being shared all the Automation engineers via GITGUB
 2. BaseClass should be available in generic libraries package.
 3. As per the Rule , Every testScripts class should extend BaseClass, so that all the configure annotation will be inherited & executed to the test scripts automatically

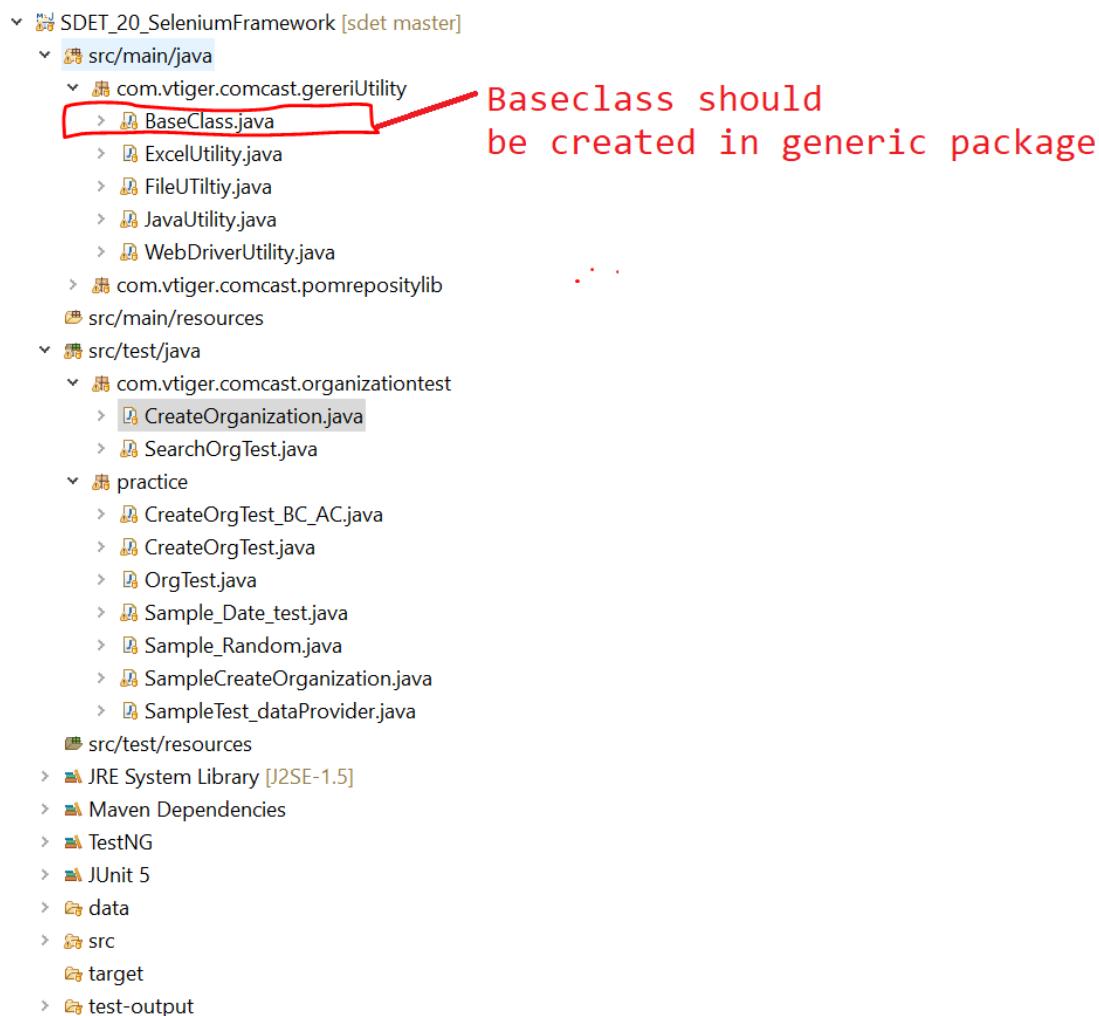
Advantages :

1. Code Reusability
2. Code Optimization
3. Modification is easy
4. Maintenance is easy
5. Test Development is faster
6. Slightly execution time is reduced

EG :



Project Structure in Eclipse



=====Base class Program=====

```

package com.vtiger.comcast.gererUtility;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.AfterSuite;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.BeforeSuite;

import com.vtiger.comcast.pomrepositorylib.Home;
import com.vtiger.comcast.pomrepositorylib.Login;

public class BaseClass {
    public WebDriver driver;
    /*Object Creation for Lib*/
    public JavaUtility jLib = new JavaUtility();
    public WebDriverUtility wLib = new WebDriverUtility();
    public FileUTiltiy fLib = new FileUTiltiy();
    public ExcelUtility eLib = new ExcelUtility();

    @BeforeSuite
    public void configBS() {
  
```

```

        System.out.println("=====connect to DB=====");

    }

    @BeforeClass
    public void configBC() {
        System.out.println("=====Launch the Browser=====");
        driver = new ChromeDriver();
        wLib.waitUntilPageLoad(driver);
        driver.manage().window().maximize();
    }

    @BeforeMethod
    public void configBM() throws Throwable {
        /*common Data*/
        String USERNAME = fLib.getPropertyKeyValue("username");
        String PASSWORD = fLib.getPropertyKeyValue("password");
        String URL = fLib.getPropertyKeyValue("url");
        String BROWSER = fLib.getPropertyKeyValue("browser");
        /* Navigate to app*/
        driver.get(URL);
        /* step 1 : login */
        Login loginPage = new Login(driver);
        loginPage.loginToApp(USERNAME, PASSWORD);
    }

    @AfterMethod
    public void configAM() {
        /*step 6 : logout*/
        Home homePage = new Home(driver);
        homePage.logout();
    }

    @AfterClass
    public void configAC() {
        System.out.println("=====Close the Browser=====");
        driver.quit();
    }

    @AfterSuite
    public void configAS() {
        System.out.println("=====close DB=====");
    }
}

```

=====Sample Test Script code using base class=====

```

package com.vtiger.comcast.organizationtest;

import org.testng.annotations.Test;

import com.vtiger.comcast.gereriuility.BaseClass;
import com.vtiger.comcast.pomrepositorylib.CreateNewOrganization;
import com.vtiger.comcast.pomrepositorylib.Home;
import com.vtiger.comcast.pomrepositorylib.OrganizationInfo;
import com.vtiger.comcast.pomrepositorylib.Organizations;

public class CreateOrganization extends BaseClass{

    @Test
    public void createOrgTest() throws Throwable {

        int randomInt = jLib.getRandomNumber();
        /*test script Data*/
        String orgName = eLib.getDataFromExcel("Sheet1", 1, 2) + randomInt;

        /*step 2 : navigate to organization*/
        Home homePage = new Home(driver);
        homePage.getOrganizationLnk().click();
    }
}

```

```

/*step 3 : navigate to "create new organization"page by click on "+" image */
Organizations orgPage = new Organizations(driver);
orgPage.getCreateOrgImg().click();

/*step 4 : create organization*/
CreateNewOrganization cno = new CreateNewOrganization(driver);
cno.createOrg(orgName);

/*step 5 : verify the successful msg with org name*/
OrganizationInfo orginfoPage = new OrganizationInfo(driver);
String actSuccesfullMg = orginfoPage.getSuccesfullMsg().getText();
if(actSuccesfullMg.contains(orgName)) {
    System.out.println(orgName + "==>created successfully");
} else {
    System.out.println(orgName + "==> not created successfully");
}

}

}

@Test
public void createOrgWithIndutriesTest() throws Throwable {
    /*test script Data*/
    int randomInt = jLib.getRandomNumber();
    String orgName = eLib.getDataFromExcel("Sheet1", 4, 2) + randomInt;
    String industriesType = eLib.getDataFromExcel("Sheet1", 4, 3);
    /*step 2 : navigate to organization*/
    Home homePage = new Home(driver);
    homePage.getOrganizationLnk().click();

    /*step 3 : navigate to "create new organization"page by click on "+" image */
    Organizations orgPage = new Organizations(driver);
    orgPage.getCreateOrgImg().click();

    /*step 4 : create organization*/
    CreateNewOrganization cno = new CreateNewOrganization(driver);
    cno.createOrg(orgName, industriesType);

    /*verify orgname & industry */
    OrganizationInfo orginfoPage = new OrganizationInfo(driver);
    String actSuccesfullMg = orginfoPage.getSuccesfullMsg().getText();
    if(actSuccesfullMg.contains(orgName)) {
        System.out.println(orgName + "==>created successfully");
    } else {
        System.out.println(orgName + "==> not created successfully");
    }

    String actIndustryType = orginfoPage.getIndutryTypeInfo().getText();
    if(actIndustryType.equals(industriesType)) {
        System.out.println(industriesType + "==>industry is verified successfully");
    } else {
        System.out.println(industriesType + "==>industry is not verified successfully");
    }
}

}

@Test
public void createOrgWithRatingTest() throws Throwable {
    /*test script Data*/
    int randomInt = jLib.getRandomNumber();
    String orgName = eLib.getDataFromExcel("Sheet1", 7, 2) + randomInt;
    String rating = eLib.getDataFromExcel("Sheet1", 7, 3);
    /*step 2 : navigate to organization*/
    Home homePage = new Home(driver);
    homePage.getOrganizationLnk().click();

    /*step 3 : navigate to "create new organization"page by click on "+" image */
    Organizations orgPage = new Organizations(driver);
    orgPage.getCreateOrgImg().click();

    /*step 4 : create organization*/
    CreateNewOrganization cno = new CreateNewOrganization(driver);
}

```

```

cno.createOrg(orgName, rating, true);

/*verify orgname & industry */
OrganizationInfo orginfoPage = new OrganizationInfo(driver);
String actSuccessfullMg = orginfoPage.getSuccessfullMsg().getText();
if(actSuccessfullMg.contains(orgName)) {
System.out.println(orgName + "==>created successfully");
}else {
System.out.println(orgName + "==> not created successfully");

}
String actRatingType = orginfoPage.getRatingTypeInfo().getText();

if(actRatingType.equals(rating)) {
System.out.println(rating + "==>industry is verified successfully");
}else {
System.out.println(rating + "==>industry is not verified successfully");
}

}
}
}
}

```

Batch Execution

- Collection of multiple test script is called batch, execute multiple test script through xml in a single click is called batch execution.
- In order to achieve batch execution, we go for Testng.xml configuration file
- TestNG xml file always start with suite xml tag followed by <test> and <classes.>
- In one xml file we can invoke N-number of testng classes, but all the classes should be present within a project.
- All the class should follow by packageNAme

EG :

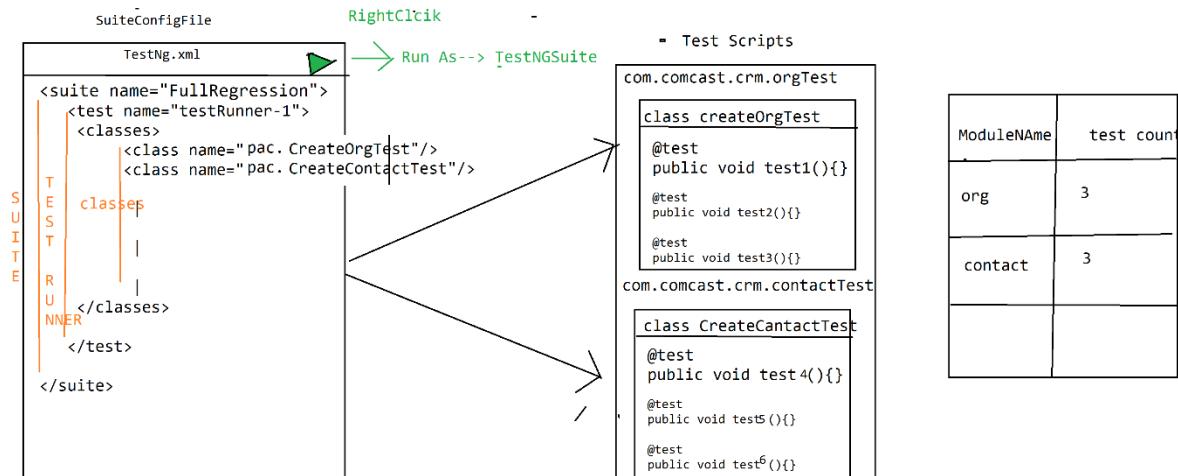
```
<class name="com.comcast.orgtest.CreateOrg"></class>
```

How to create testNG xml file automatically through eclipse?

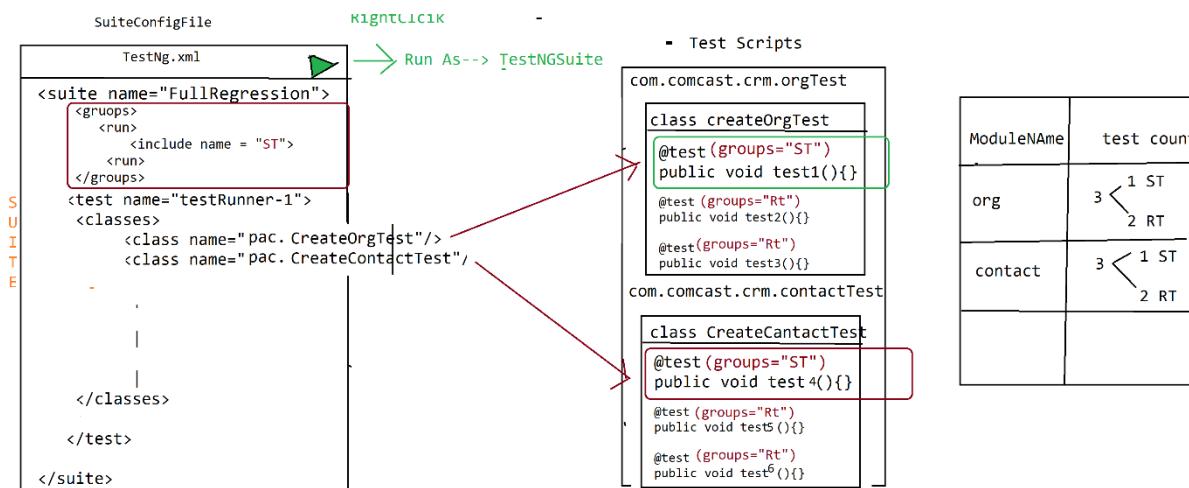
Select all the testNG classes or packages -> right click-> select ➔ testNG ➔ and click on “Convert to testing” and ➔ click on finish.

- ⇒ Automatically you will get the testng.xml file with in the project
- ⇒ In order to edit xml File ➔ double click on testing.xml ➔ click on “Source”

Batch Execution : [Full regression testing]



Grouping Execution:



- Collection of similar test scripts across the testing classes is called grouping Execution
 - In order achieve grouping execution , each & every test script should have group name, group name will be written along with annotation
 - In grouping execution, all configure annotation should have group name , other wise those annotation will not participate in grouping execution like `@BeforeSuite @BeforeClass , @BeforeMethod etc`
- EG :

```

@BeforeSuite(groups = {"smokeTest", "regressionTest"})
public void configBS() {
    System.out.println("=====Execute BeforeSuite=====");
}

```

Smoke Test:

```

@Test(groups={"smokeTest"})
public void createCustomerTest()
{
    System.out.println("execute createCustomerTest");
}

```

```
}
```

```
@Test(groups={"regressionTest"})
    public void modifyCustomerTest()
{
    System.out.println("execute modifyCustomerTest");
}
```

- In order to invoke grouping execution should declare Group Key in testing.xml file & group key should be declared before <test>, after <suite> tag

Smoke Test:

```
<suite name="Suite">
    <groups>
        <run>
            <include name="smokeTest"/>
        </run>
    </groups>
    <test name="Test">
        <classes>
            <class name="pac1.ProjectAndCustomerTest"/>
            <class name="pac2.ReportTest"/>

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

- We can invoke multiple groupkey in one XML File

```
<suite name="Suite">
    <groups>
        <run>
            <include name="smokeTest"/>
            <include name="regressionTest"/>
        </run>
    </groups>
    <test name="Test">
        <classes>
            <class name="pac1.ProjectAndCustomerTest"/>
            <class name="pac2.ReportTest"/>

        </classes>
    </test>
```

→ One test can have multiple Group name

```
@Test(groups={"regressionTest","smokeTest"})
    public void modifyCustomerTest()
{
    System.out.println("execute modifyCustomerTest");
}
```

Regional Regression Test:

→ To execute particular test cases across the Suite is called regional regression testing

→ in Real time , impact area is given by developer / test lead , based on that idea XML will be created

→ Whenever we want execute particular @test method inside class , we go for <method> & <include>

```
<?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="com.comcast.crm.contacttest.CreateContactTest">
                <methods>
                    <include name="createdContactTest"/>
                </methods>
            </class>
            <class name="com.comcast.crm.orgtest.CreateOrgTest">
                <methods>
                    <include name="createdOrgWithRatingTest"/>
                </methods>
            </class>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

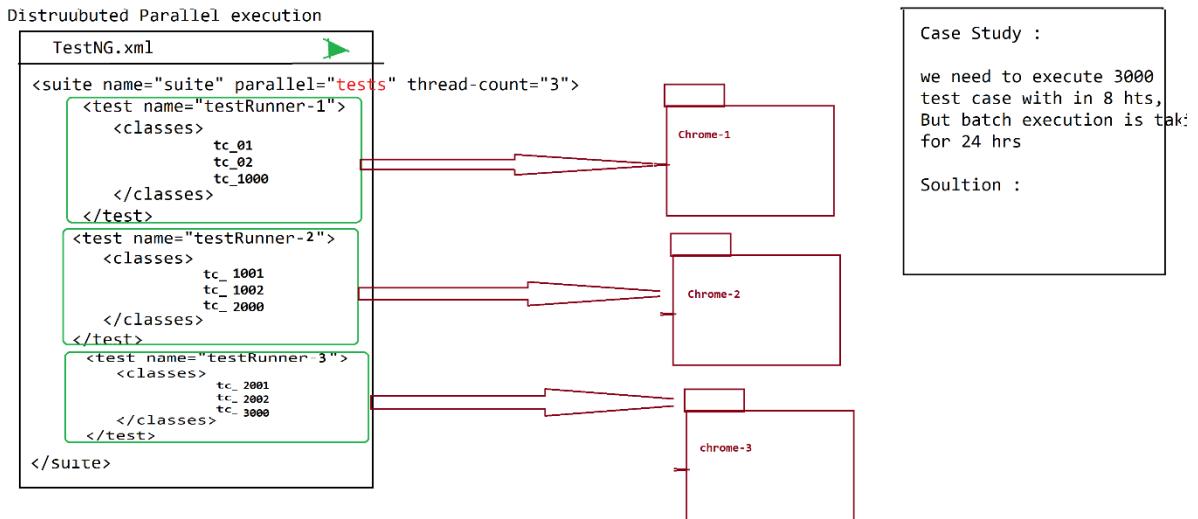
Parallel Execution:

There are 3 types of Parallel Execution

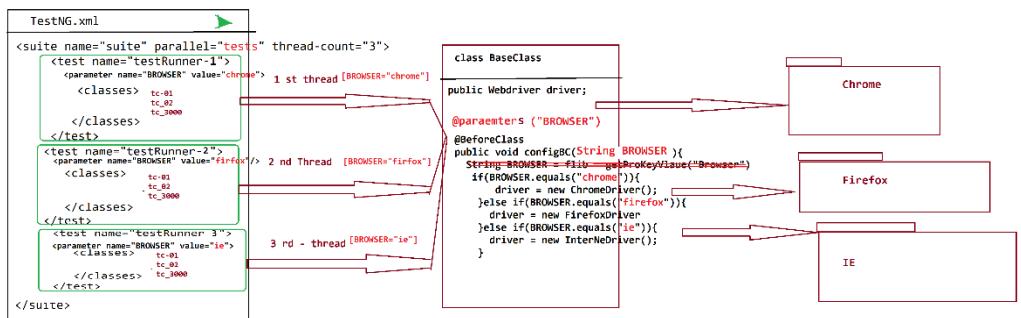
1. Distributed Parallel execution:
2. Cross browser parallel execution
3. Cross platform parallel execution

1. Distributed Parallel execution:

- Distribute the test case across the multiple test runner & execute, each <test> Test runner in parallel is called Distributed Parallel execution
- ⇒ we reduce the suite execution time, so that we can get the result early
- ⇒ in order to achieve parallel execution we should enable parallel="tests" & thread-count=5 in <suite>, then create multiple test runner & distribute the testcase
- ⇒ maximum thread count is 5
- ⇒ Thread count should be same as number of <test> testRunner



2 Cross browser parallel execution / Browser Compatibility testing



1. Execute same set of testcase in different Browser parallel is called cross browser testing
2. To achieve cross browser testing we should use <parameter> in XML file & @parameters annotation inside the testScript

3. <parameter> is used to specify the browser data for each <test>

EG :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel= "tests" thread-count="2">
    <test name="Test-Runner-Chrome">
        <parameter name="BROWSER" value="chrome"/>
        <classes>
            <class name="com.vtiger.comcast.organizationtest.CreateOrganization"/>
            <class name="com.vtiger.comcast.organizationtest.SearchOrgTest"/>
        </classes>
    </test> <!-- Test -->

    <test name="Test-Runner-Firefox">
        <parameter name="BROWSER" value="firefox"/>
        <classes>
            <class name="com.vtiger.comcast.organizationtest.CreateOrganization"/>
            <class name="com.vtiger.comcast.organizationtest.SearchOrgTest"/>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

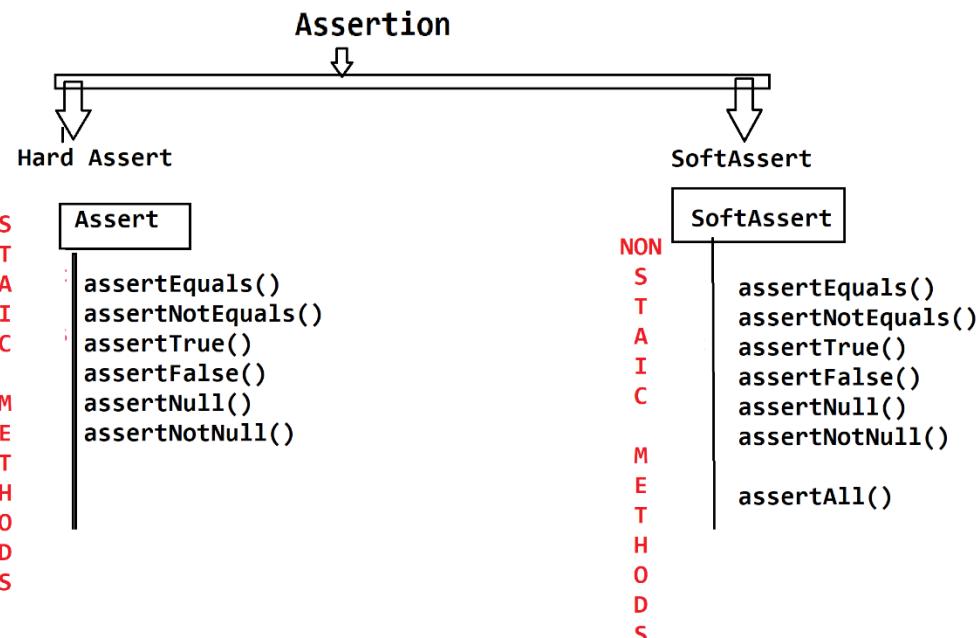
4. @parameters annotation will be used to receive the data (Eg : browser , platform etc) from the XML file to test Scripts [Like BaseClass]

Note : Replace below annotation inside the balseclass

```
@Parameters("BROWSER")
@BeforeClass
public void configBC(String BROWSER) {
    System.out.println("=====Launch the Browser=====");
    if(BROWSER.equals("chrome")) {
        driver = new ChromeDriver();
    }else if(BROWSER.equals("firefox")) {
        driver = new FirefoxDriver();
    }else if(BROWSER.equals("ie")) {
        driver = new InternetExplorerDriver();
    }else {
        driver = new ChromeDriver();
    }
    wLib.waitUntilPageLoad(driver);
    driver.manage().window().maximize();
}
```

Assertion / CheckPoint

- ⇒ Assertion is a feature available in TestNG used to validate test scripts expected results
- ⇒ As per the Rule of the automation every expected result should be verified with Assert statements, because java “if else “statement will not have capability to fail the testNG test Scripts
- ⇒ There are 2 types of Assertions in TESTNG



Hard Assertion	Soft Assertion
All methods are static in nature	All methods are non-static in nature
It does not allow further execution of test if the line containing hard assert gets failed.	Next steps would be executed even if the line containing soft assertion gets failed.
Whole test case gets failed if at least 1 hard assert fails.	AssertAll() extra lines of code are required to track the fail status.
To verify mandatory fields we go for hard assert	To verify non mandatory fields we go for soft assert

HardAssert

Whenever hardAssert method fails, testNG generate AssertError exception & stop the current test execution & continue execution with remaining test

```
@Test  
public void createCustomerTest(){
```

```

        System.out.println("step_1");
        System.out.println("step_2");
        Assert.assertEquals("A", "B");
        System.out.println("step_3");
        System.out.println("step_4");
    }
    @Test
    public void modifyCustomerTest(){
        System.out.println("=====");
        System.out.println("step_1");
        System.out.println("step_2");
        System.out.println("step_3");
    }
}

```

Out Put

```

step_1
step_2
=====
step_1
step_2
step_3
PASSED: modifyCustomerTest
FAILED: createCustomerTest
java.lang.AssertionError: expected [B] but found [A]

```

Soft Assert

Whenever softAssert mtd fails , testNG Generate AssertionError exception & continue execution with remaining steps of same testScript

```

public void createCustomerTest(){
    System.out.println("step_1");
    System.out.println("step_2");
    SoftAssert s = new SoftAssert();
    s.assertEquals("A", "B");
    System.out.println("step_3");
    s.assertEquals("X", "Y");
    System.out.println("step_4");
    s.assertAll();
}
@Test
public void modifyCustomerTest(){
    System.out.println("=====");
    System.out.println("step_1");
    System.out.println("step_2");
    System.out.println("step_3");
    System.out.println("step_4");
}

```

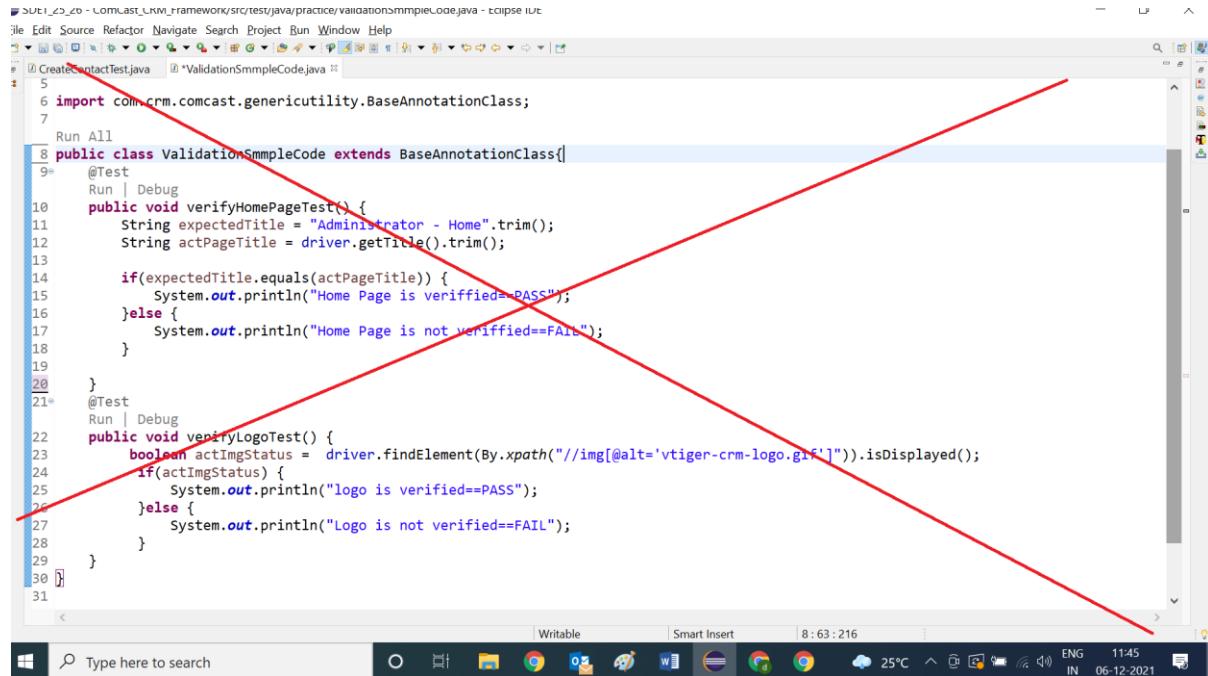
Output

```

step_1
step_2
step_3
step_4
=====
step_1
step_2
step_3
step_4
PASSED: modifyCustomerTest
FAILED: createCustomerTest
java.lang.AssertionError: The following asserts failed:
    expected [B] but found [A],
    expected [Y] but found [X]

```

Assertion usage in Real Time



The screenshot shows the Eclipse IDE interface with a Java file named `CreateContactTest.java` open. The code contains two test methods: `verifyHomePageTest()` and `verifyLogoTest()`. The `verifyHomePageTest()` method compares the expected title ("Administrator - Home") with the actual page title. The `verifyLogoTest()` method checks if an image element is displayed based on its alt attribute. A large red 'X' is drawn across the entire code block, indicating that the code is incorrect or failing.

```

SUT1_2020 - ComLast_LKM_Framework/src/test/java/practice/validationSimpleCode.java - eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
CreateContactTest.java *ValidationSimpleCode.java
5
6 import com.crm.comcast.genericutility.BaseAnnotationClass;
7
8 public class ValidationSimpleCode extends BaseAnnotationClass{
9     @Test
10    public void verifyHomePageTest() {
11        String expectedTitle = "Administrator - Home".trim();
12        String actPageTitle = driver.getTitle().trim();
13
14        if(expectedTitle.equals(actPageTitle)) {
15            System.out.println("Home Page is verified==PASS");
16        }else {
17            System.out.println("Home Page is not verified==FAIL");
18        }
19    }
20    @Test
21    public void verifyLogoTest() {
22        boolean actImgStatus = driver.findElement(By.xpath("//img[@alt='vtiger-crm-logo.gif']")).isDisplayed();
23        if(actImgStatus) {
24            System.out.println("Logo is verified==PASS");
25        }else {
26            System.out.println("Logo is not verified==FAIL");
27        }
28    }
29 }
30
31

```

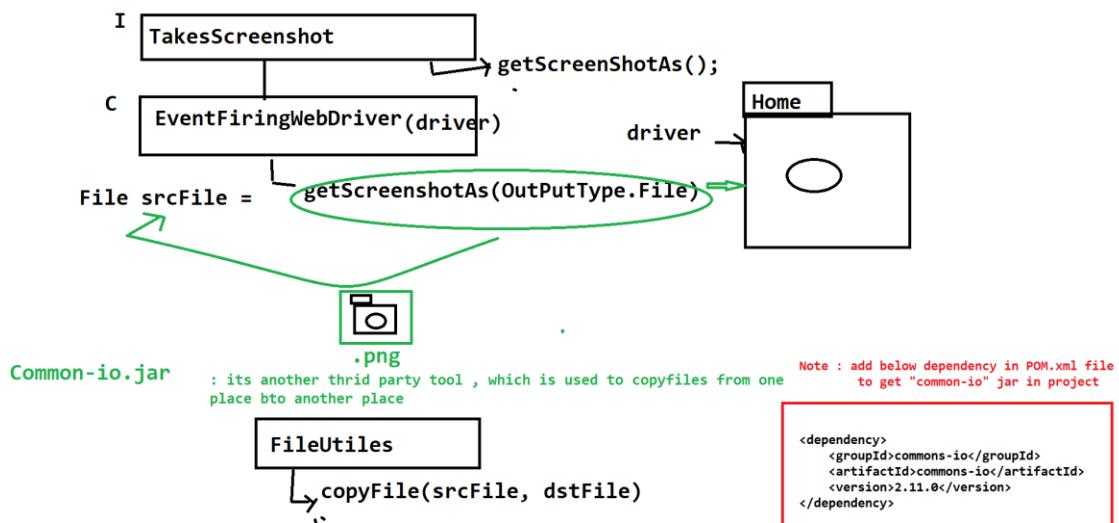
The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** SDET_20_SeleniumFramework/src/test/java/practice/SampleTest.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar with icons for file operations.
- Package Explorer:** Shows the project structure with packages like practice, org, and testng, and files such as CreateOrgTest_BC_AC.java, OrgTestJava, SampleTest.java, etc.
- Code Editor:** Displays Java code for a TestNG test class named SampleTest. The code includes imports for org.openqa.selenium.* and org.testng.annotations.*. It contains two test methods: verifyHomePage() and verifyLogoInHomePage(). Both methods use System.out.println for logging, capture the page title, and assert that the actual title equals the expected title.
- Bottom Status Bar:** Shows the current time (18:53), date (25-08-2021), and system information (ENG IN 29°C).

Advantages of Assertion:

- ⇒ It's used to fail the TESTNG test scripts
- ⇒ It's used for test scripts validation
- ⇒ It's generate “AssertErrorException” & reason of the failure + failed line number whenever test is failed
- ⇒ We can compare any 2 primitive variable or array or Collection or MAP in single line

Screen Shot



WebDriver Code to take a Screenshot

```

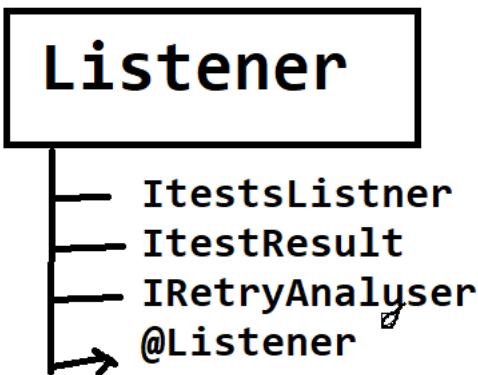
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.support.events.EventFiringWebDriver;
import org.testng.Assert;
import org.testng.annotations.Test;
import com.vtiger.comcast.gereriuility.BaseClass;

public class SampleTest extends BaseClass {
    @Test
    public void verifyHomePage(Method mtd) throws Throwable{
        System.out.println(mtd.getName());
        String currentTestName = mtd.getName();
        System.out.println("=====Test START=====");
        EventFiringWebDriver edriver = new EventFiringWebDriver(driver);
        File srcFile = edriver.getScreenshotAs(OutputType.FILE);
        File dstFile = new File("./Screenshot/"+currentTestName+".png");
        FileUtils.copyFile(srcFile, dstFile);
        System.out.println("=====Test END=====");
    }
}

```

Listener

- ⇒ Listen is feature available in TESTNG , which is used to capture runtime events during execution & perform appropriate action based on eventtype



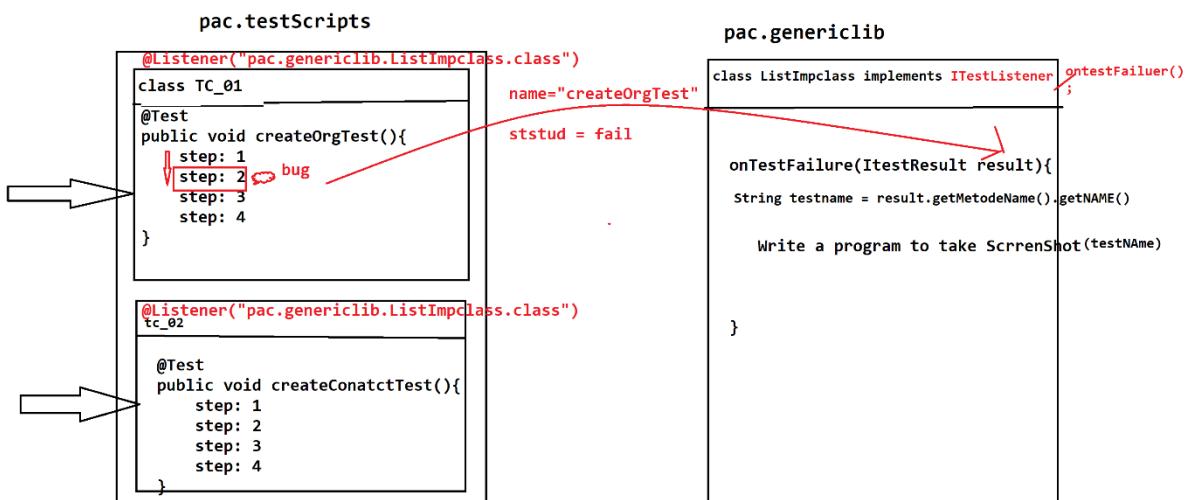
ITestListener:

- ⇒ This is a special feature in testNG which enables the user to take run time events whenever the test scripts fail/pass. Implementation class for ITestListener is mandatory to use Listener feature
- ⇒ @Listener is testing annotation , which is used to monitor the test execution in the runtime & generate a runtime event to Listener Implementation class , if test is pass / fail
- ⇒ @Listener annotation will be declared in every testScripts class before class definition block
- ⇒ ListImplementation class helps us to receive the failure events from the @Listener & perform appropriate actions

Advantages of Listener :

- ⇒ We can use Listener to take a screenshot for the failed test case, when we execute in bulk
- ⇒ We can also use Listener for connect to DB , Launch browser & login precondition program
- ⇒ We can also use Listener for Extend Report Configuration

Listener Implementation class :



SDET_20 - SDET_20_SeleniumFramework/src/main/java/com/vtiger/comcast/gereriUtility/LisImpClass.java - Eclipse IDE

```

1 package com.vtiger.comcast.gereriUtility;
2
3 import java.io.File;
4
5 public class LisImpClass implements ITestListener{
6
7     public void onTestFailure(ITestResult result) {
8
9         String testName = result.getMethod().getMethodName();
10        System.out.println(testName + "=====Execute & i am Listnening=====");
11
12        EventFiringWebDriver eDriver = new EventFiringWebDriver(BaseClass.sDriver);
13        File srcFile = eDriver.getScreenshotsAs(OutputType.FILE);
14        try {
15            FileUtils.copyFile(srcFile, new File("./screenshot/"+testName+".png"));
16        } catch (IOException e) {
17            e.printStackTrace();
18        }
19    }
20
21 }

```

Sample test for Listener :

SDET20 - SDET_20_SeleniumFramework/src/test/java/practice/SampleTest.java - Eclipse IDE

```
1 package practice;
2
3 import org.testng.Assert;
4 import org.testng.annotations.Listeners;
5 import org.testng.annotations.Test;
6
7 import com.vtiger.comcast.gereriUtility.BaseClass;
8
9 @Listeners(com.vtiger.comcast.gereriUtility.LisImpClass.class)
10 public class SampleTest extends BaseClass {
11
12     @Test
13     Run | Debug
14     public void aaaaa() throws Throwable{
15
16         System.out.println("=====Test START=====");
17
18         Assert.assertEquals(false, true);
19
20         System.out.println("=====Test END=====");
21     }
22 }
```

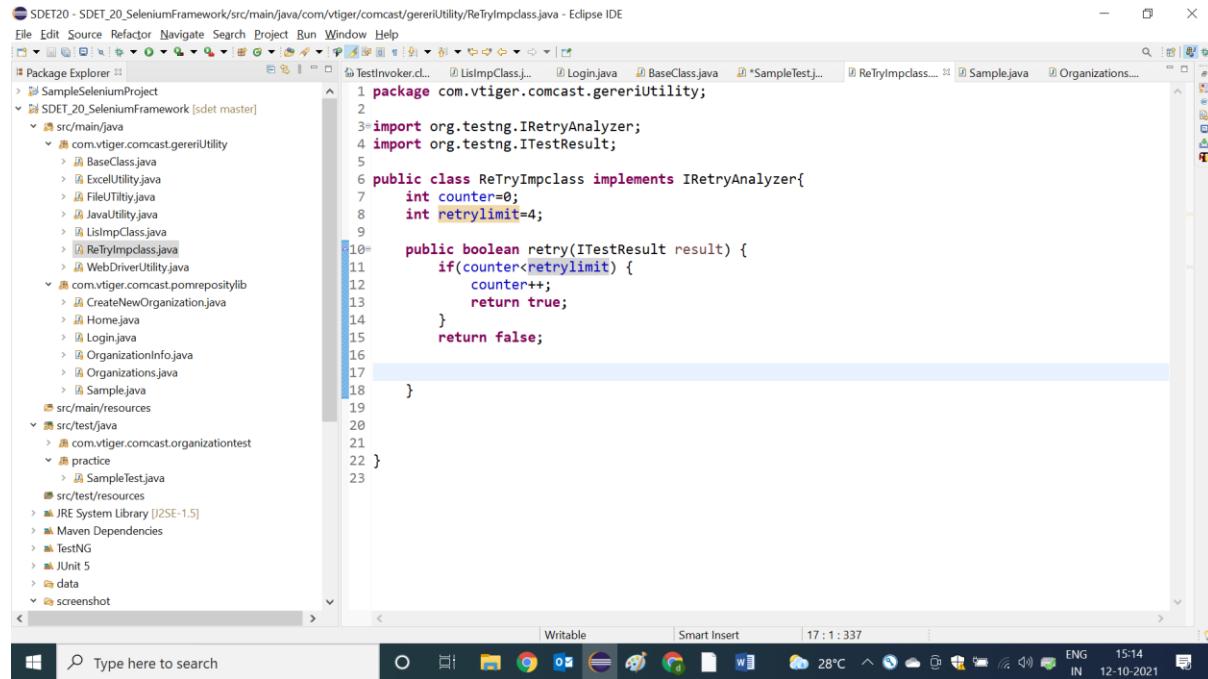
How to use Listener in TestNG.xml

SDET20 - SDET_20_SeleniumFramework/testng.xml - Eclipse IDE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
    <listeners>
        <listener class-name="com.vtiger.comcast.gereriUtility.LisImpClass"/>
    </listeners>
    <test thread-count="5" name="Test">
        <classes>
            <class name="com.vtiger.comcast.organizationtest.CreateOrganization"/>
            <class name="com.vtiger.comcast.organizationtest.SearchOrgTest"/>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

RetryAnalyzer:

- ⇒ This feature of TestNG helps the user to rerun the test script when ever test is getting failed
- ⇒ In order to use this feature we have to implements RetryAnalyzer interface & override retry method
- ⇒ Inside the retry method, we should specify the upper limit so that test script will get executed specified limit when test is getting failed.



The screenshot shows the Eclipse IDE interface with the title bar "SDET20 - SDET_20_SeleniumFramework/src/main/java/com/vtiger/comcast/gereriUtility/ReTryImpclass.java - Eclipse IDE". The left side features the "Package Explorer" view displaying the project structure under "SampleSeleniumProject". The right side is the code editor with the following Java code:

```
1 package com.vtiger.comcast.gereriUtility;
2
3 import org.testng.IRetryAnalyzer;
4 import org.testng.ITestResult;
5
6 public class ReTryImpclass implements IRetryAnalyzer{
7     int counter=0;
8     int retrylimit=4;
9
10    public boolean retry(ITestResult result) {
11        if(counter<retrylimit) {
12            counter++;
13            return true;
14        }
15        return false;
16    }
17
18 }
19
20
21
22 }
```

The code implements the `IRetryAnalyzer` interface with a single method `retry`. It initializes a counter to 0 and a retry limit to 4. Inside the `retry` method, it checks if the counter is less than the retry limit. If true, it increments the counter and returns `true`, indicating the test should be retried. Otherwise, it returns `false`.

```
1 package practice;
2
3 import java.io.File;
4
5 Run All
6 public class SampleTest extends BaseClass{
7
8     @Test(retryAnalyzer = com.vtiger.comcast.gereriUtility.ReTryImpclass.class)
9     public void amazontest() throws IOException {
10
11         Assert.assertEquals("A", "B");
12
13     }
14
15 }
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
```

When ever we execute 1000 test scripts in batch, out of 1000 / 100 test scripts got failed , next time when a get new build , I wanted execute only failed test scripts , then is your approach ?

After the batch execution, refresh the project & go to test-output folder which is created inside the projectFolder , then execute “test-failed.xml” file

- ⇒ Testng-failed.xml file is created automatically by testing-tool itself for every failed execution reports .