**Generic Libraries / Utility**

**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 /Actions

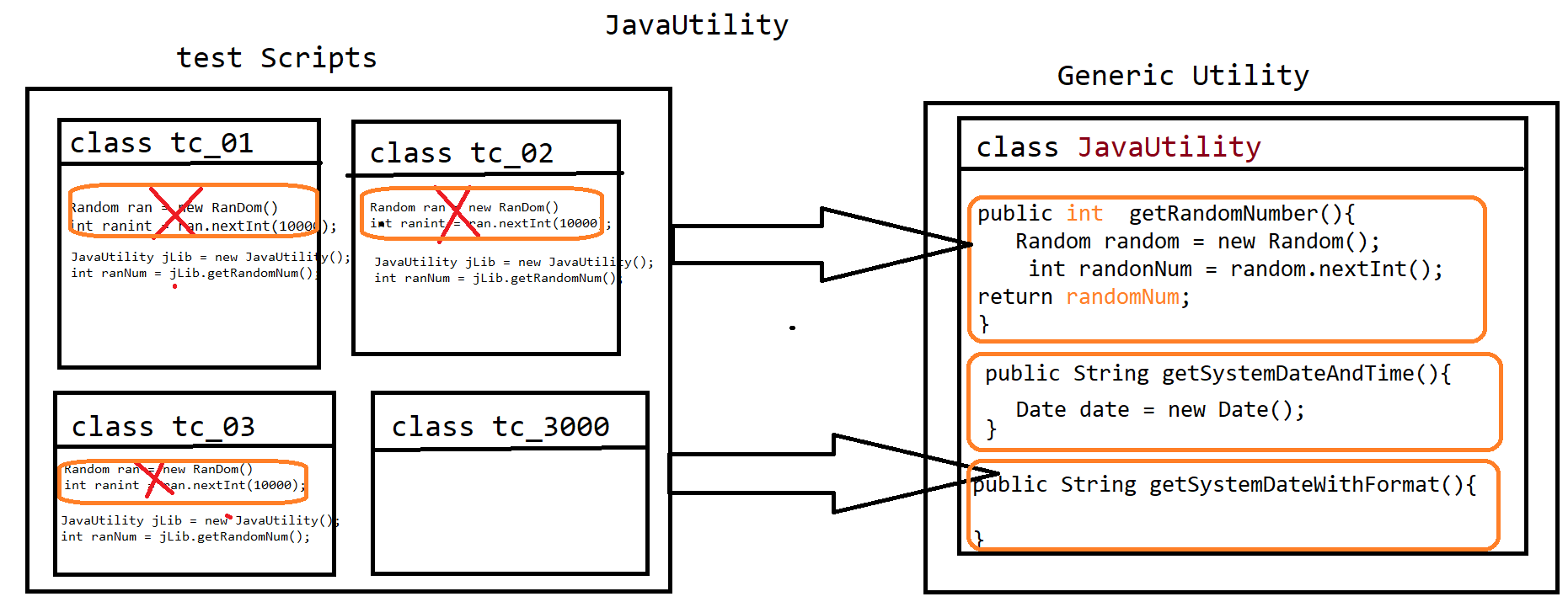
🡺The methods which can be used to any application is called Generic/common methods

**What is the advantages of Generic components?**

* Reusability of code
* Code Optimization
* Test script development is faster
* Test 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

=====================Code for Java Utility===============================

**package** com.comcast.genericutlity;

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.Random;

/\*\*

\* contains java specific utility

\*

\* **@author** Deepak

\*

\*/

**public** **class** JavaUtlity {

Random random = **new** Random();

Date dateObj = **new** Date();

SimpleDateFormat sim = **new** SimpleDateFormat("yyyy-MM-dd");

/\*\*

\* generate random number with in the limit of 40000 for very invocation

\*

\* **@return**

\*/

**public** **int** getRandomNumber() {

**int** ranInt = random.nextInt(4000);

**return** ranInt;

}

/\*\*

\* used to get the system current date in "yyyy-MM-dd " format

\*

\* **@return**

\*/

**public** String getDate() {

String date = sim.format(dateObj);

**return** date;

}

/\*\*

\* used to get the required date in "yyyy-MM-dd "

\* format requiredDateCount is positive number , it provides upcoming date based numeric count

\* if requiredDateCount is negative number , it provides previous date based numeric count

\*

\* **@param** requiredDateCount

\*/

**public** String getDate(**int** requiredDateCount) {

Calendar cal = Calendar.*getInstance*();

cal.add(Calendar.***DATE***, requiredDateCount);

Date date = cal.getTime();

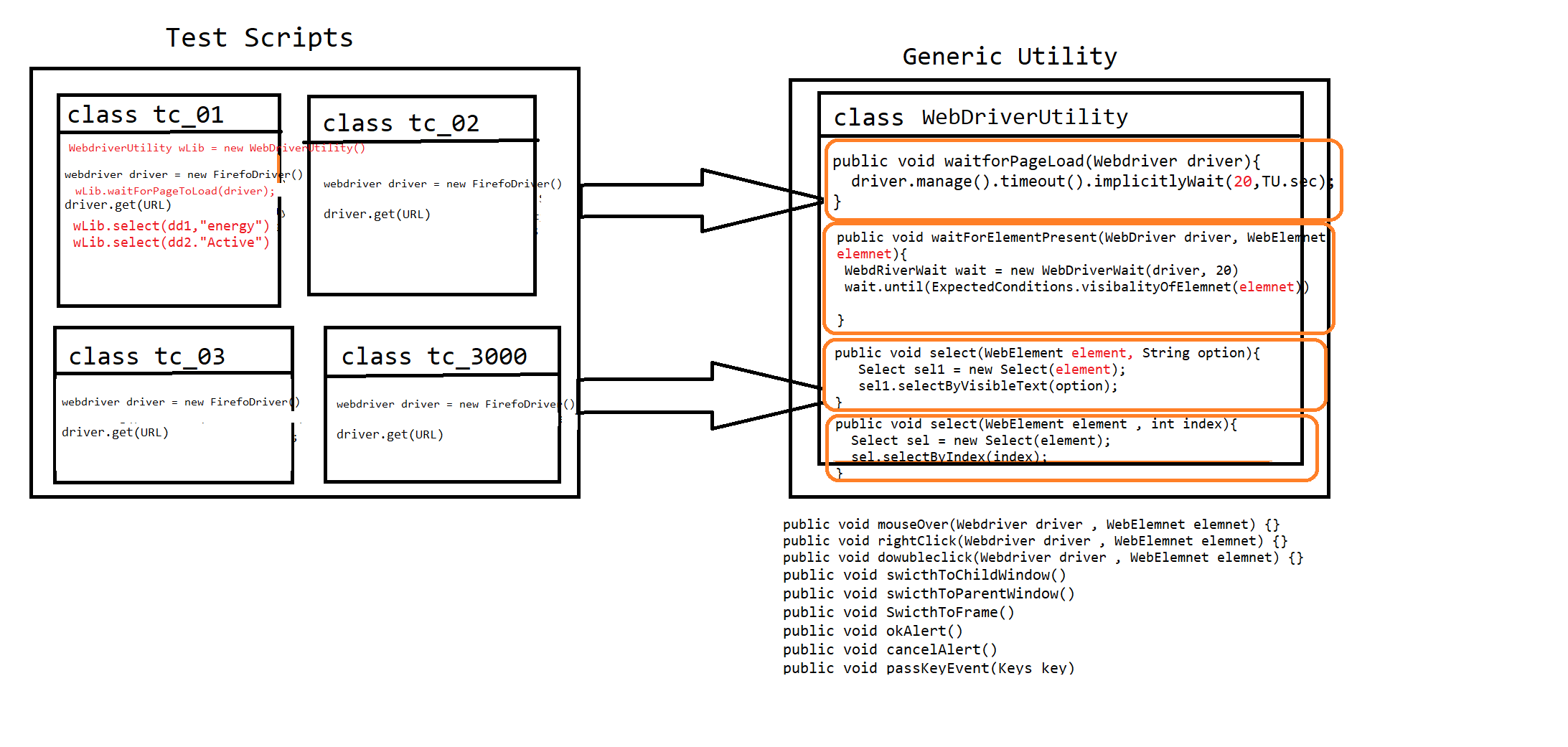
String rdata = sim.format(date);

**return** rdata;

}

}

1. WebDriver Utility Libraries



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

==========Code======================

package com.comcast.genericutlity;

import java.time.Duration;

import java.util.Iterator;

import java.util.Set;

import org.openqa.selenium.Alert;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.Keys;

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;

/\*\*

\* contains all web browser related actions which includes mouse move , select , switchToWindow , wait etc

\* @author Deepak

\*

\*/

public class WebActionUtility {

FileUtlity fLib = new FileUtlity();

int TIMEOUT;

public WebActionUtility() throws Throwable {

String filepath = fLib.getFilePathFromPropertiesFile("projectConfigDataFilePath");

String STIMEOUT = fLib.getDataFromProperties(filepath, "TimeOut");

TIMEOUT = Integer.parseInt(STIMEOUT);

}

/\*\*

\* it's an implicitly wait Always wait for Element in DOM document & release the control if element available

\* @param driver

\* @throws Throwable

\*/

public void waitForElementInDOM(WebDriver driver) throws Throwable {

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(TIMEOUT));

}

/\*\*

\* it's an Explicitly wait Always wait for Page to be loaded & available in GUI

\* @param driver

\* @param partailPageURL

\* @throws Throwable

\*/

public void waitForPage(WebDriver driver , String partailPageURL) throws Throwable {

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(TIMEOUT));

wait.until(ExpectedConditions.urlContains(partailPageURL));

}

/\*\*

\* it's an Explicitly wait Always wait for Page to be loaded & available in GUI

\* @param driver

\* @param partailPageURL

\* @throws Throwable

\*/

public void waitForElement(WebDriver driver , WebElement element) throws Throwable {

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(TIMEOUT));

wait.until(ExpectedConditions.visibilityOf(element));

}

/\*\*

\*

\* @param element

\* @throws Throwable

\*/

public void waitAndClick(WebElement element) throws Throwable

{

int count = 0;

while(count<TIMEOUT) {

try {

element.click();

break;

}catch(Throwable e){

Thread.sleep(1000);

count++;

}

}

}

/\*\*

\* wait for the element and type the data

\* @param element

\* @param data

\* @throws InterruptedException

\*/

public void waitAndType(WebElement element, String data) throws InterruptedException

{

int count = 0;

while(count<20) {

try {

element.sendKeys(data);

break;

}catch(Throwable e){

Thread.sleep(1000);

count++;

}

}

}

/\*\*

\* used to Switch to Any Window based on Window Title

\* @param driver

\* @param partialWindowTitle

\*/

public void swithToWindow(WebDriver driver , String partialWindowTitle) {

Set<String> set = driver.getWindowHandles();

Iterator<String> it = set.iterator();

while (it.hasNext()) {

String wID = it.next();

driver.switchTo().window(wID);

String currentWindowTitle = driver.getTitle();

if(currentWindowTitle.contains(partialWindowTitle)) {

System.out.println(partialWindowTitle + "Switch to Window is passed--!");

break;

}

}

}

/\*\*

\* used to Switch to Any Window based on Window URL

\* @param driver

\* @param partialWindowTitle

\*/

public void swithToWindowBasedOnURL(WebDriver driver , String partialWindowURL) {

Set<String> set = driver.getWindowHandles();

Iterator<String> it = set.iterator();

while (it.hasNext()) {

String wID = it.next();

driver.switchTo().window(wID);

String currentWindowTitle = driver.getCurrentUrl();

if(currentWindowTitle.contains(partialWindowURL)) {

System.out.println(partialWindowURL + "Switch to Window is passed--!");

break;

}

}

}

/\*\*

\* used to Switch to Alert Window & click on OK button

\* @param driver

\*/

public void swithToAlertWindowAndAccpect(WebDriver driver ,String expectedMsg) {

Alert alt = driver.switchTo().alert();

if(alt.getText().trim().equalsIgnoreCase(expectedMsg.trim())) {

System.out.println("Alert Message is verified");

}else {

System.out.println("Alert Message is not verified");

}

alt.accept();

}

/\*\*

\* used to Switch to Alert Window & click on Cancel button

\* @param driver

\*/

public void swithToAlertWindowAndCancel(WebDriver driver, String expectedMsg) {

Alert alt = driver.switchTo().alert();

if(alt.getText().equals(expectedMsg)) {

System.out.println("Alert Message is verified");

}else {

System.out.println("Alert Message is not verified");

}

alt.dismiss();

}

/\*\*

\* used to Switch to Frame Window based on index

\* @param driver

\* @param index

\*/

public void swithToFrame(WebDriver driver , int index) {

driver.switchTo().frame(index);

}

/\*\*

\* used to Switch to Frame Window based on id or name attribute

\* @param driver

\* @param id\_name\_attribute

\*/

public void swithToFrame(WebDriver driver , String id\_name\_attribute) {

driver.switchTo().frame(id\_name\_attribute);

}

/\*\*

\* used to select the value from the dropDwon based on index

\* @param driver

\* @param index

\*/

public void select(WebElement element , int index) {

Select sel = new Select(element);

sel.selectByIndex(index);

}

/\*\*

\* used to select the value from the dropDwon based on value / option avlaible in GUI

\* @param element

\* @param value

\*/

public void select(WebElement element , String text) {

Select sel = new Select(element);

sel.selectByVisibleText(text);

}

/\*\*

\* used to place mouse cursor on specified element

\* @param driver

\* @param elemnet

\*/

public void mouseOverOnElement(WebDriver driver , WebElement elemnet)

{

Actions act = new Actions(driver);

act.moveToElement(elemnet).perform();

}

/\*\*

\* used to right click on specified element

\* @param driver

\* @param elemnet

\*/

public void rightClickOnElement(WebDriver driver , WebElement elemnet)

{

Actions act = new Actions(driver);

act.contextClick(elemnet).perform();

}

/\*\*

\*

\* @param driver

\* @param javaScript

\*/

public void executeJavaScript(WebDriver driver , String javaScript) {

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeAsyncScript(javaScript, null);

}

/\*\*

\* pass enter Key appertain in to Browser

\* @param driver

\*/

public void passEnterKey(WebDriver driver) {

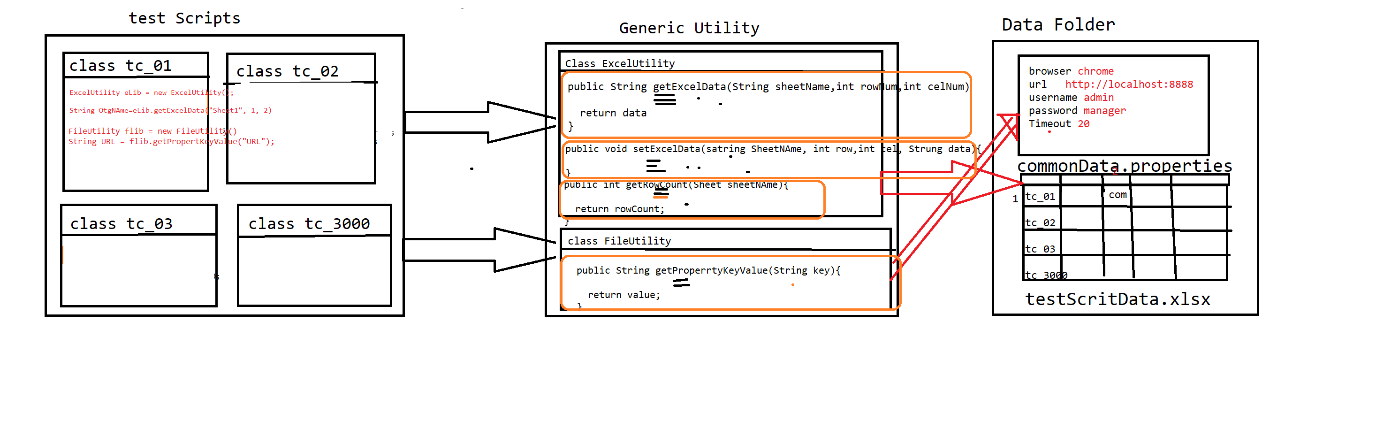
Actions act = new Actions(driver);

act.sendKeys(Keys.ENTER).perform();

}

}

1. 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 & .propertes file

We go for ExcelUtility & FileUtlity

* 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

=================FileUtility===============================

package com.comcast.genericutlity;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.util.Properties;

/\*\*

\* used to read data from external files like text , proprties

\* @author Deepak

\*

\*/

public class FileUtlity {

/\*\*

\* used to get the filepaths from the /config/filepath.properties FILE

\* @param key

\* @return value

\* @throws Throwable

\*/

public String getFilePathFromPropertiesFile(String key) throws Throwable {

FileInputStream fis = new FileInputStream("./config/filepath.properties");

Properties pObj = new Properties();

pObj.load(fis);

String value = pObj.getProperty(key);

return value;

}

/\*\*

\* used to get the key : value from the any properties based on file Path parsmeter

\* @param filePath

\* @param key

\* @return value

\* @throws Throwable

\*/

public String getDataFromProperties(String filePath, String key) throws Throwable {

FileInputStream fis = new FileInputStream(filePath);

Properties pObj = new Properties();

pObj.load(fis);

String value = pObj.getProperty(key);

return value;

}

}

==========Excel Utility Code=======================

**package** com.comcast.genericutlity;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**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;

/\*\*

\* used to perfrom Workbook releated operations like read , write , getcount etc

\* **@author** Deepak

\*

\*/

**public** **class** ExcelUtlity {

/\*\*

\* read the data from excel based row & column index

\* **@param** filePath

\* **@param** sheetName

\* **@param** rowNum

\* **@param** celNum

\* **@return**

\* **@throws** Throwable

\* **@throws** IOException

\*/

**public** String getDataFromExcel(String filePath , String sheetName, **int** rowNum , **int** celNum) **throws** Throwable, IOException {

FileInputStream fis1 = **new** FileInputStream(filePath);

Workbook wb = WorkbookFactory.*create*(fis1);

Sheet sheet = wb.getSheet(sheetName);

Row row = sheet.getRow(rowNum);

String data = row.getCell(celNum).toString();

wb.close();

**return** data;

}

/\*\*

\* read the date based on testId and Required Column name

\* **@param** filePath

\* **@param** sheetName

\* **@param** testId

\* **@param** columnHeader

\* **@return**

\* **@throws** Throwable

\* **@throws** IOException

\*/

**public** String getDataFromExcelBasedTestId(String filePath, String sheetName , String testId, String columnHeader ) **throws** Throwable, IOException {

FileInputStream fis1 = **new** FileInputStream(filePath);

Workbook wb = WorkbookFactory.*create*(fis1);

Sheet sheet = wb.getSheet(sheetName);

**int** rowCount = sheet.getLastRowNum();

**int** testRowNum = 0;

String actTestID="";

String actColHeaderName="";

String data="";

**for**(**int** i=0; i<=rowCount; i++) {

**try** { actTestID = sheet.getRow(i).getCell(0).toString();} **catch** (Exception e) {}

**if**(actTestID.equalsIgnoreCase(testId)) {

**break**;

}

testRowNum++;

}

**int** testColNum =0;

**int** celCountforTest = sheet.getRow(testRowNum-1).getLastCellNum();

**for**(**int** j=0; j<celCountforTest; j++) {

**try** {actColHeaderName = sheet.getRow(testRowNum-1).getCell(j).toString(); } **catch** (Exception e) {}

**if**(actColHeaderName.equalsIgnoreCase(columnHeader)) {

**break**;

}

testColNum++;

}

**try** {data = sheet.getRow(testRowNum).getCell(testColNum).toString();}**catch** (Exception e) {}

**return** data;

}

/\*\*

\* used to get the maximum used row count in required Sheet

\* **@param** filePath

\* **@param** sheetName

\* **@return**

\* **@throws** Throwable

\*/

**public** **int** getRowCount(String filePath, String sheetName) **throws** Throwable {

FileInputStream fis1 = **new** FileInputStream(filePath);

Workbook wb = WorkbookFactory.*create*(fis1);

Sheet sheet = wb.getSheet(sheetName);

**int** rowCount = sheet.getLastRowNum();

**return** rowCount;

}

/\*\*

\*

\* **@param** filePath

\* **@param** sheetName

\* **@param** testId

\* **@param** columnHeader

\* **@param** data

\*/

**public** **void** setDataExcel(String filePath, String sheetName , **int** rowNum, **int** celNum ,String data) **throws** Throwable {

FileInputStream fis = **new** FileInputStream(filePath);

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(filePath);

wb.write(fos);

wb.close();

}

}

================================Data base Utility Code==================

Data base uitilty is implemented using JDBC , which is used to connect to any database using java code,

Below utility contains 4 methods used for below operations

1. connectDB: used for db connection
2. closeDB : used for close db connection
3. execyteQuery : used to execute select query
4. execyteUpdate : used to execute non select query
5. executeQuerryAndVerify : used to execute query & verify the data in db in specified column

package com.crm.comcast.genericutility;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import com.mysql.jdbc.Driver;

public class DataBaseUtility {

static Driver driver;

static Connection connection;

static ResultSet result;

/\*\*

\* This method will perform the mysql database connection

\* @throws SQLException

\*/

public void connectDB() throws SQLException {

try {

driver=new Driver();

DriverManager.registerDriver(driver);

connection=DriverManager.getConnection(IConstants.JDBC\_URL\_String,IConstants.JDBC\_USERNAME,IConstants.JDBC\_PASSWORD);

} catch (SQLException e) {

e.printStackTrace();

}

}

/\*\*

\* This method will close the mysql database

\* @throws SQLException

\*/

public void closeDB() throws SQLException {

try {

connection.close();

}catch (Exception e) {

}

}

/\*\*

\* This method will execute the querry

\* @param query

\* @return

\* @throws Throwable

\*/

public ResultSet execyteQuery(String query) throws Throwable {

result = connection.createStatement().executeQuery(query);

return result;

}

/\*\*

\* This method will execute the querry

\* @param query

\* @return

\* @throws Throwable

\*/

public int execyteUpdate(String query) throws Throwable {

int result = connection.createStatement().executeUpdate(query);

return result;

}

/\*\*

\* This method will execute querry based on querry and it will verify the data.

\* @param querry

\* @param columnName

\* @param expectedData

\* @return

\* @throws Throwable

\*/

public boolean executeQuerryAndVerify(String querry,int cloumnIndex,String expectedData) throws Throwable {

boolean flag=false;

result=connection.createStatement().executeQuery(querry);

while(result.next()) {

if(result.getString(cloumnIndex).equals(expectedData)) {

flag=true;

break;

}

}

if(flag) {

System.out.println(expectedData+"==>Data is verified in the data base table");

return flag;

}else {

System.out.println(expectedData+"==>data is not verified in the database");

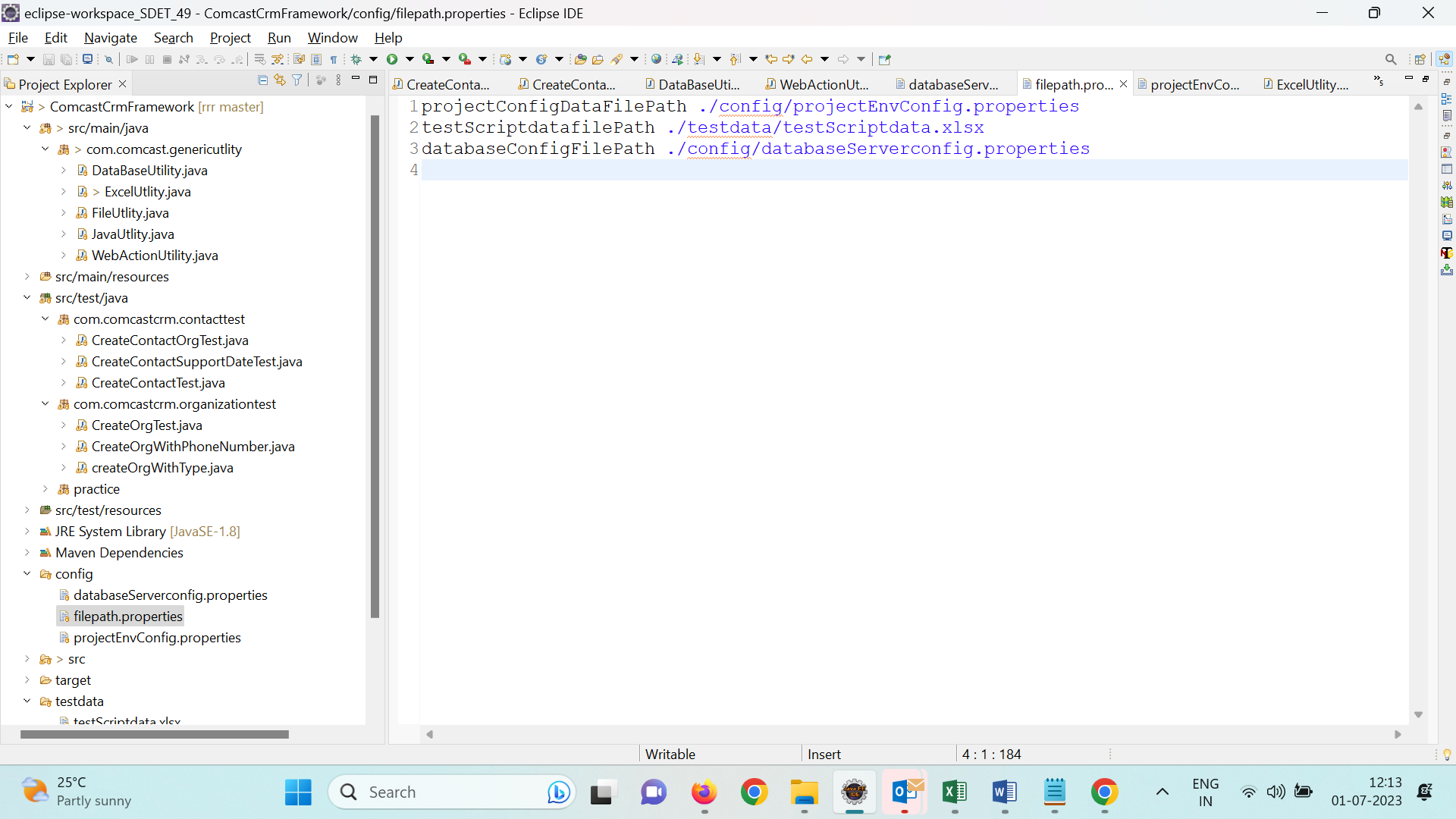
return flag;

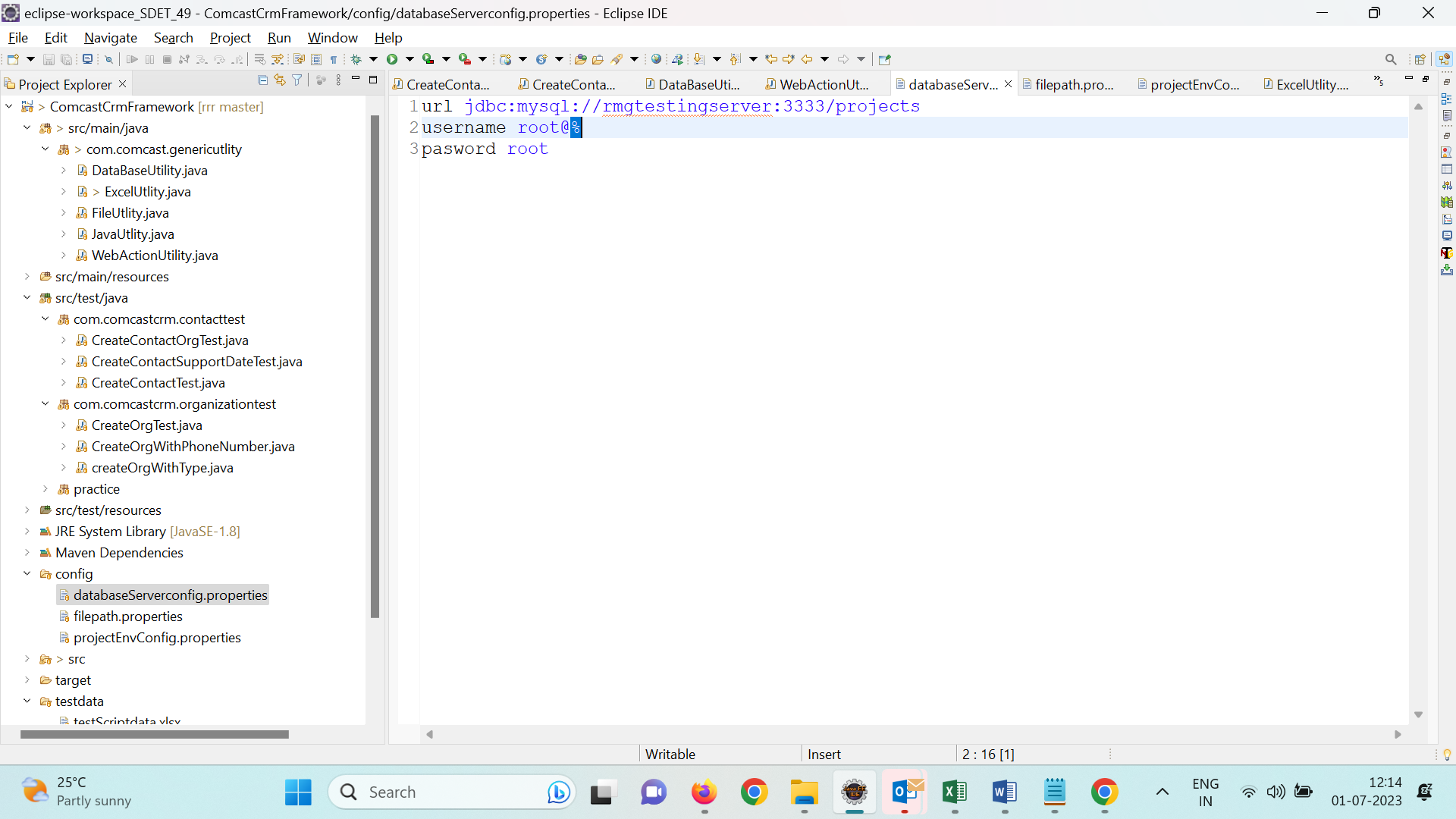
}

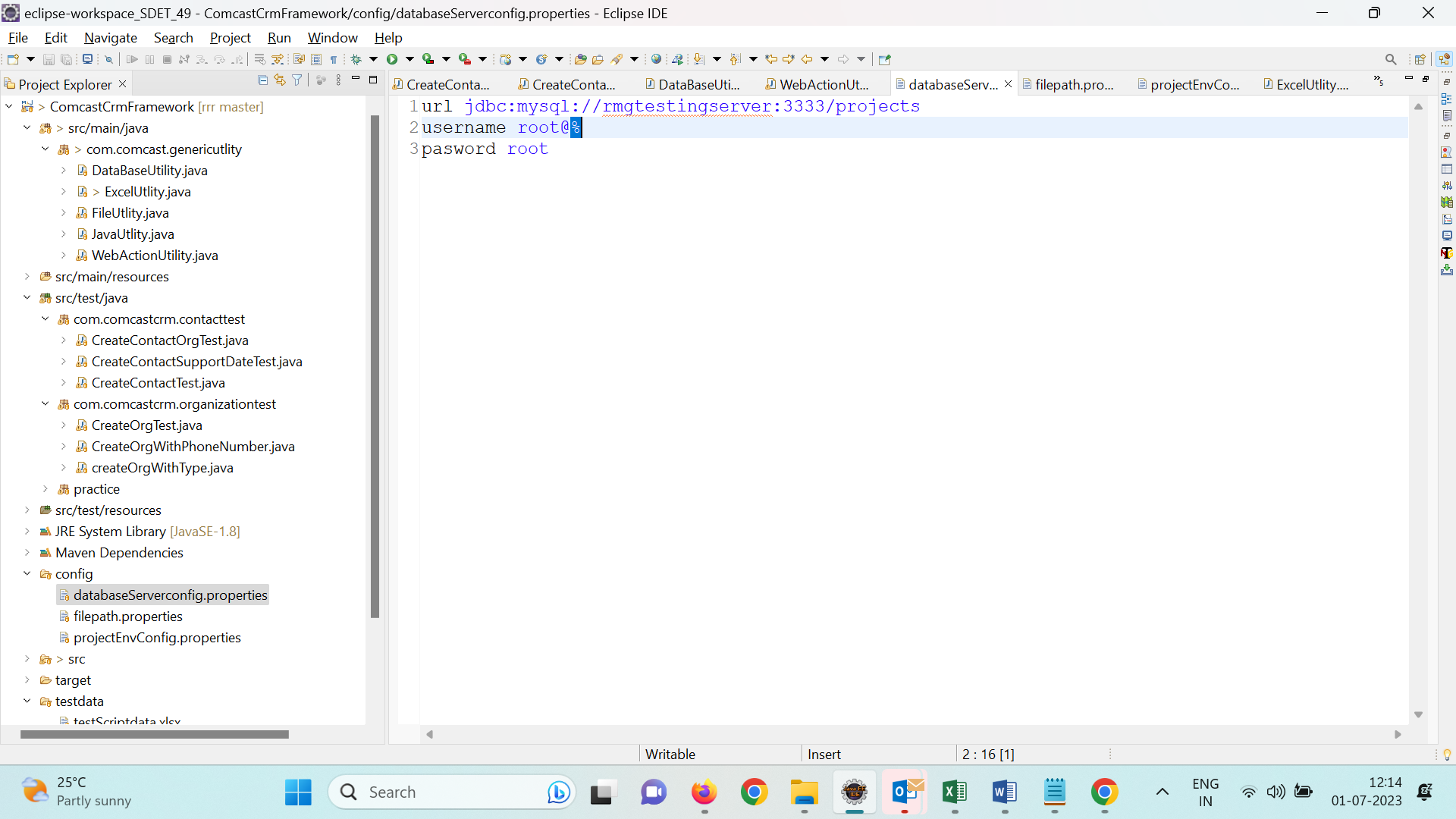
}

}

}







============test Name : CreateContactWithOrgTest =================

package com.comcastcrm.contacttest;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.edge.EdgeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.Test;

import com.comcast.genericutlity.ExcelUtlity;

import com.comcast.genericutlity.FileUtlity;

import com.comcast.genericutlity.JavaUtlity;

import com.comcast.genericutlity.WebActionUtility;

public class CreateContactOrgTest {

@Test

public void createOrgTest() throws Throwable{

/\*create Object for utlity \*/

FileUtlity fLib = new FileUtlity();

ExcelUtlity eLib = new ExcelUtlity();

JavaUtlity jLib = new JavaUtlity();

WebActionUtility wLib = new WebActionUtility();

/\*get the FILE PATH\*/

String ENV\_FILE\_PATH = fLib.getFilePathFromPropertiesFile("projectConfigDataFilePath");

String TEST\_SCRIPT\_EXCEL\_FILE\_PATH = fLib.getFilePathFromPropertiesFile("testScriptdatafilePath");

/\*Read the common data\*/

String BROWSER = fLib.getDataFromProperties(ENV\_FILE\_PATH, "browser");

String URL = fLib.getDataFromProperties(ENV\_FILE\_PATH, "url");

String USERNAME = fLib.getDataFromProperties(ENV\_FILE\_PATH, "username");

String PASSWORd = fLib.getDataFromProperties(ENV\_FILE\_PATH, "password");

/\*test script data\*/

int randomNum = jLib.getRandomNumber();

String orgName = eLib.getDataFromExcelBasedTestId(TEST\_SCRIPT\_EXCEL\_FILE\_PATH, "contact", "tc\_06", "OrgName") +"\_"+ randomNum;

String lastName = eLib.getDataFromExcelBasedTestId(TEST\_SCRIPT\_EXCEL\_FILE\_PATH, "contact", "tc\_06", "ContactLastName") +"\_"+ randomNum;

WebDriver driver = null;

/\*step 1 : login to app\*/

if(BROWSER.equalsIgnoreCase("chrome")) {

driver = new ChromeDriver();

}else if(BROWSER.equalsIgnoreCase("firefox")) {

driver = new FirefoxDriver();

}else if(BROWSER.equalsIgnoreCase("edge")) {

driver = new EdgeDriver();

}else {

driver = new ChromeDriver();

}

wLib.waitForElementInDOM(driver);

driver.get(URL);

driver.findElement(By.name("user\_name")).sendKeys(USERNAME);

driver.findElement(By.name("user\_password")).sendKeys(PASSWORd);

driver.findElement(By.id("submitButton")).click();

/\*step 2 : navigate to Org page\*/

driver.findElement(By.linkText("Organizations")).click();

/\*step 3 : navigate to create Org page\*/

driver.findElement(By.xpath("//img[@alt='Create Organization...']")).click();

/\*step 4 : create a new org\*/

driver.findElement(By.name("accountname")).sendKeys(orgName);

driver.findElement(By.xpath("//input[@title='Save [Alt+S]']")).click();

/\*verify expected result\*/

String actHeader = driver.findElement(By.className("dvHeaderText")).getText();

if(actHeader.contains(orgName)) {

System.out.println(orgName+ "is verified PASS");

}else {

System.out.println(orgName+ "is not verified FAIL");

}

wLib.waitForElement(driver, driver.findElement(By.xpath("//img[@alt='Create Contact...']")));

/\*step 5 : navigate to Org page\*/

driver.findElement(By.linkText("Contacts")).click();

/\*step 6 : navigate to create Org page\*/

driver.findElement(By.xpath("//img[@alt='Create Contact...']")).click();

/\*step 4 : create a new org\*/

driver.findElement(By.name("lastname")).sendKeys(lastName);

driver.findElement(By.xpath("//input[@name='account\_name']/following-sibling::img")).click();

//switch to child window

wLib.swithToWindowBasedOnURL(driver, "module=Accounts");

driver.findElement(By.id("search\_txt")).sendKeys(orgName);

driver.findElement(By.name("search")).click();

driver.findElement(By.xpath("//a[text()='"+orgName+"']")).click();

//switch to parent window window

wLib.swithToWindowBasedOnURL(driver, "module=Contacts");

driver.findElement(By.xpath("//input[@title='Save [Alt+S]']")).click();

/\*verify expected result\*/

actHeader = driver.findElement(By.className("dvHeaderText")).getText();

if(actHeader.contains(lastName)) {

System.out.println(lastName+ "is verified PASS");

}else {

System.out.println(lastName+ "is not verified FAIL");

}

String actOrgName = driver.findElement(By.xpath("//td[@id='mouseArea\_Organization Name']/a")).getText();

if(actOrgName.contains(orgName)) {

System.out.println(orgName+ "is verified PASS");

}else {

System.out.println(orgName+ "is not verified FAIL");

/\*step 4 : logout\*/

WebElement ele = driver.findElement(By.xpath("//img[@src='themes/softed/images/user.PNG']"));

wLib.mouseOverOnElement(driver, ele);

driver.findElement(By.linkText("Sign Out")).click();

driver.close();

}

}

}