1. POM Framework:

In Page Object Model framework, Object identification code and validation code are separated



In this approach we create individual classes for every page web elements with identification code and then we create individual functions for each Webelement to return identification value

Advantages of POM:

-In future or next versions if the object properties are changed, then we need to update only WebElement identification code (i.e. Object Repository package classes only)

-it will increase Readability of the script

-it will reduce duplication of code

-it is easy for maintainability

In POM framework we use PageFactory class and @FindBy annotation

PageFactory:

Using this class we can initialize Webelement locator value classes in required class

Syntax:

ClassName Obj= PageFactory.initElements(driver, ClassName.class);

@FindBy

Using this annotation we can assign Webelement locator value

Syntax:

@FindBy(locatorname= locator\_value)

WebElement ele;

==================================================

ex:

create script to validate login functionality in <http://rediffmail.com> application using POM Framework with PageFactory annotation @FindBy

Procedure:

Step1:

create 2 packages

package-1: "object.repository"

package-2: "test.cases"

Step 2:

create 2 non-executable classes in "object.repository" package

class-1: RediffHomePg

Class-2: RediffLoginPg

Step 3:

assign home page webelement locator value to the object

using @FindBy annotation in "RediffHomePg" class

syntax:

@FindBy(Locator="Locator value")

WebElement obj;

Step 4:

create method/function to return obj from "RediffHomePg" class

syntax:

public WebElement methodname(){

return(obj);

}

Script in "RediffHomePg" class

-------------------------

public class RediffHomePg {

//Assign locator value

@FindBy(linkText="Sign in")

WebElement signin;

//to return obj

public WebElement signinObj(WebDriver driver){

return(signin);

}

}

Step 5: assign Login page webelement locator values to the objects

using @FindBy annotation in "RediffLoginPg" class and create

method/functions to return each obj

class

script in "RediffLoginPg" class

--------------------------------

public class RediffLoginPg {

//Assign locator value

@FindBy(name="login")

WebElement emailId;

//Assign locator value

@FindBy(name="passwd")

WebElement pwd;

//Assign locator value

@FindBy(name="proceed")

WebElement go;

//to return obj

public WebElement emailObj(WebDriver driver){

return(emailId);

}

public WebElement pwdObj(WebDriver driver){

return(pwd);

}

public WebElement goObj(WebDriver driver){

return(go);

}

}

Step 6: create class in "test.cases" package to validate login functionality

By initializing Webelement locator values from “object.repository” package classes using “PageFactory” class

Syntax:{to initialize objects from "object.repository" package classes}

ClassName Obj= PageFactory.initElements(driver, ClassName.class);

public class RediffLoginValidation {

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

WebDriver driver= new FirefoxDriver();

driver.get("http://rediffmail.com");

driver.manage().window().maximize();

//to initialize objects from "object.repository" package classes

RediffHomePg RHP= PageFactory.initElements(driver, RediffHomePg.class);

RediffLoginPg RLP= PageFactory.initElements(driver, RediffLoginPg.class);

RHP.signinObj(driver).click();

RLP.emailObj(driver).sendKeys("Madhukar");

RLP.pwdObj(driver).sendKeys("Mercury");

RLP.goObj(driver).click();

Thread.sleep(3000);

String pgTitle= driver.getTitle();

if (pgTitle.contains("Inbox")){

System.out.println("Successful login operation");

}

else{

System.out.println("Unsuccessful login operation");

}

}

}

Properties file:

It is also called as configuration properties file

Where we can maintain reusable test data in properties file like appln path, user id, pwd, locator values...etc

Each parameter in properties file is stored as pair of strings, in key and value format, where each key is one line

Ex:

Uid= Livetech

To read Properties file we use Properties class object

Following are the methods provided by Properties class

1. load(): to load the properties file in current test case
2. getProperty(): to read key value from properties file

Procedure to work on Properties file:

step1:

Create new folder in java project with name as "properties"

Navigation:

Select java project in eclipse

Right click on mouse

New

Select "Folder"

Enter Folder name (i.e. Properties)

Click on Finish

step2:

Create properties file under this new folder with name as "inputdata.properties"

Navigation:

Select "Properties" folder under java project

Right click on mouse

New

Select "File"

Enter file name (inputdata.properties)

Click on Finish

Step 3: assign required data to the keys in properties file

ex:

baseUrl=https://opensource-demo.orangehrmlive.com/

userId=Admin

pwd=admin123

Ex: create script to perform login operation in Orange HRM project by taking test data from properties file

step 4:

Create class to perform login operation using data from properties file

**package** datadriven.framework;

**import** java.io.FileInputStream;

**import**java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.Properties;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**publicclass** OrangeHRM {

**publicstaticvoid** main(String[] args) **throws** IOException {

//to specify Properties file location

FileInputStream fi= **new** FileInputStream("./Properties//inputdata.properties");

//create Object for properties class

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

prop.load(fi);

//to read key values from properties file

String autURL=prop.getProperty("baseUrl");

String userName=prop.getProperty("userId");

String pwd=prop.getProperty("pwd");

//To initialize browser

System.*setProperty*("webdriver.chrome.driver","D:\\SelResource\\chromedriver.exe");

WebDriver driver= **new** ChromeDriver();

driver.get(autURL);

driver.manage().window().maximize();

//to perform login operation

driver.findElement(By.*name*("txtUsername")).sendKeys(userName);

driver.findElement(By.*name*("txtPassword")).sendKeys(pwd);

driver.findElement(By.*name*("Submit")).click();

}

}

3. Data Driven Framework: (Re-testing)

-------------------------------------------

Whenever we parameterize test script with multiple set of test data is called Data driven Framework

Passing the test data to the script during runtime is called Parameterization

Parameterization will increase reusability of the script

In general to parameterize test script with multiple set of values we prefer workbook (.xls/.xlsx) files

Whereas WebDriver does not support excel directly

Using third party drivers like "Apache POI API" we can work on workbook

Note:

jxl jars will support only .xls file due to that reason which is not preferred

\*\*We can read and write on excel file with help of Java IO package and Apache POI library.

POI -(Poor Obfuscation Implementation )jars

Downloading Apache poi

URL:http://poi.apache.org/download.html

Click on "poi-bin-4.0.1-20181217.zip" link under "Binary Distribution"section

Click on any one of the .zip link under "BACKUP SITES"

Extract the zip file

Configure all the jar files to java project

in Eclipse

Note:

external jar files, OOxml folder jars and Lib folder jars

Excel file structure:

------------------------

Excel file -->workbook -->Sheet -->Row -->Cell

Classes and Interfaces in Apache POI

-----------------------------------

Interfces --> XLSXClasses:

-------- -------

Workbook--> XSSFWorkbook

Sheet --> XSSFSheet

Row --> XSSFRow

Cell --> XSSFCell

Note:

XSSF -->XML SpreadSheet Format

HSSF -->Horrible SpreadSheet Format

note:

to work on xls file we can use following classes

HSSFWorkbook, HSSFSheet, HSSFRow and HSSFCell

To handle files we can use following classes from Java IO package:

FileInputStream --> to read data from file

FileOutputStream -->to write data to file

Ex: write program to enter data in Excel file

//to specify File location

FileOutputStream fo=new FileOutputStream("D:\\demodata.xlsx");

//to create workbook

XSSFWorkbook workbook= new XSSFWorkbook();

//create sheet

XSSFSheet sheet=workbook.createSheet("Students");

//to specify Row-1

XSSFRow row1=sheet.createRow(0);

//to enter data in a column

row1.createCell(0).setCellValue("St\_Name");

row1.createCell(1).setCellValue("Subject");

//to specify Row-2

XSSFRow row2=sheet.createRow(1);

//to enter data in a column

row2.createCell(0).setCellValue("Rajesh");

row2.createCell(1).setCellValue("Selenium");

//to specify Row-3

XSSFRow row3=sheet.createRow(2);

//to enter data in a column

row3.createCell(0).setCellValue("Gowtham");

row3.createCell(1).setCellValue("UFT");

workbook.write(fo);

System.out.println("write excel is completed");

Ex: write program to read data from Excel

//to specify file location

FileInputStream fi= new FileInputStream("D:\\demodata.xlsx");

//to specify workbook

XSSFWorkbook workbook= new XSSFWorkbook(fi);

//to specify worksheet

XSSFSheet sheet=workbook.getSheet("Students");

//to find number of rows

int rows=sheet.getLastRowNum();

int cols=sheet.getRow(0).getLastCellNum();

System.out.println("Number of rows are: "+(rows+1));

System.out.println("Number of columns are: "+cols);

for(int r=0; r<= rows; r++) {

XSSFRow row=sheet.getRow(r);

for(int col=0; col<cols;col++) {

XSSFCell cell= row.getCell(col);

String myVal=cell.toString();

System.out.println(myVal);

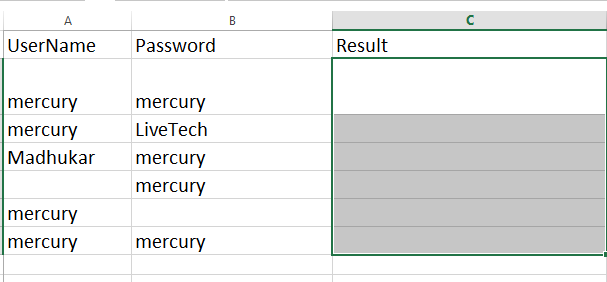
}

}

Ex: create script to validate login functionality in Mercury tours applicationby using test data from excel file using concept of Data Driven Framework

Test data in Excel file (D:\MercuryLoginData.xlsx)

Sheet:Sheet1



procedure:

Step 1: identify common operations in Excel file to perform DDT and createindividual methods to perform those operations in one class (i.e. XLUtils)

1. To find number of rows data availability

2. To find number of cells in each row

3. To read data from cells

4. To write data into cells

Create Class (i.e. XLUtils.java) with following static methods:

method-1: setExcelFile (to specify working File and Sheet )

method-2: getRowCount (To find number of Rows in a specified sheet)

method-3: getCellCount (to find number of cells in a specified Row)

method-4: getCellData (to read data)

method-5: setCellData (to enter data)

Script in XLUtils.java class

----------------------------

package ddt.framework;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import org.apache.poi.ss.usermodel.DataFormatter;

import org.apache.poi.xssf.usermodel.XSSFCell;

import org.apache.poi.xssf.usermodel.XSSFRow;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class XLUtils {

public static FileInputStream fi;

public static FileOutputStream fo;

public static XSSFWorkbook wb;

public static XSSFSheet ws;

public static XSSFRow row;

public static XSSFCell cell;

//to specify working File and Sheet

public static void setExcelFile(String xlfile,String xlsheet) throws Exception {

try {

FileInputStream ExcelFile = new FileInputStream(xlfile);

wb = new XSSFWorkbook(ExcelFile);

ws = wb.getSheet(xlsheet);

} catch (Exception e){

throw (e);

}

}

//To find number of Rows data availability in a specified sheet

public static int getRowCount(String xlfile,String xlsheet) throws IOException

{

fi=new FileInputStream(xlfile);

wb=new XSSFWorkbook(fi);

ws=wb.getSheet(xlsheet);

int rowcount=ws.getLastRowNum();

wb.close();

fi.close();

return rowcount;

}

//to find number cells in a specified Row

public static int getCellCount(String xlfile,String xlsheet,int rownum) throws IOException

{

fi=new FileInputStream(xlfile);

wb=new XSSFWorkbook(fi);

ws=wb.getSheet(xlsheet);

row=ws.getRow(rownum);

int cellcount=row.getLastCellNum();

wb.close();

fi.close();

return cellcount;

}

//to read cell value

public static String getCellData(String xlfile,String xlsheet,int rownum,int colnum) throws IOException

{

fi=new FileInputStream(xlfile);

wb=new XSSFWorkbook(fi);

ws=wb.getSheet(xlsheet);

row=ws.getRow(rownum);

cell=row.getCell(colnum);

String data;

try

{

DataFormatter formatter = new DataFormatter();

String cellData = formatter.formatCellValue(cell);

return cellData;

}

catch (Exception e)

{

data="";

}

wb.close();

fi.close();

return data;

}

//to set value

public static void setCellData(String xlfile,String xlsheet,int rownum,int colnum,String data) throws IOException

{

fi=new FileInputStream(xlfile);

wb=new XSSFWorkbook(fi);

ws=wb.getSheet(xlsheet);

row=ws.getRow(rownum);

cell=row.createCell(colnum);

cell.setCellValue(data);

fo=new FileOutputStream(xlfile);

wb.write(fo);

wb.close();

fi.close();

fo.close();

}

}

Step 2:create script to validate login operation in Mercury tours applicationby using methods from XLUtils class

System.setProperty("webdriver.chrome.driver", "D:\\SelResource\\chromedriver.exe");

WebDriver driver= new ChromeDriver();

driver.get("http://newtours.demoaut.com");

driver.manage().window().maximize();

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

String path="D:\\MercuryLoginData.xlsx";

XLUtils.setExcelFile(path, "Sheet1");

int rows=XLUtils.getRowCount(path, "Sheet1");

for(int i=1; i<=rows; i++) {

String username=XLUtils.getCellData(path, "Sheet1", i, 0);

String pwd=XLUtils.getCellData(path, "Sheet1", i, 1);

//to perform login operation

driver.findElement(By.name("userName")).sendKeys(username);

driver.findElement(By.name("password")).sendKeys(pwd);

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

Thread.sleep(8000);

//to read page title

String pgTitle=driver.getTitle();

if(pgTitle.equals("Find a Flight: Mercury Tours:")) {

System.out.println("Test passed");

XLUtils.setCellData(path, "Sheet1", i, 2, "Successful login operation");

driver.findElement(By.linkText("SIGN-OFF")).click();

}

else {

System.out.println("Test Failed");

XLUtils.setCellData(path, "Sheet1", i, 2, "Unsuccessful login operation");

}

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

}

}