03-MAR-2020

-----------

3. DATA DRIVEN FRAMEWORK : (Re-testing)

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

-- Whenever we parameterize test script with multiple set of test data is called as "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

-->\*\* Where as WebDriver does not support excel directly.

(There are no Classes or not Methods in WD to support 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 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 & lib folder jars

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

EXCEL FILE STRUCTURE

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

Excel file --> Workbook --> Sheet --> Row --> Cell

File

---> Workbook

---> Sheet

---> Row

---> Columns (Cell)

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

JRE libs

--- (Java IO Package)

FileInputStream ----> To read data

FileOutputSream ----> To write data

To represent Excel file structure "XSSF"

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

Classes and Interfaces in Apache POI

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

Interfaces ----> XLSX Classes:

Workbook ----> XSSF Workbook

Sheet ----> XSSF Sheet

Row ----> XSSF Row

Cell ----> XSSF Cell

Note: XSSF ----> XML Spread Sheet Format

HSSF ----> Horrible Spread Sheet Format // It is used for .xls

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

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 into 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();

}

}

==================================================================END OF CLASS====================================================================================