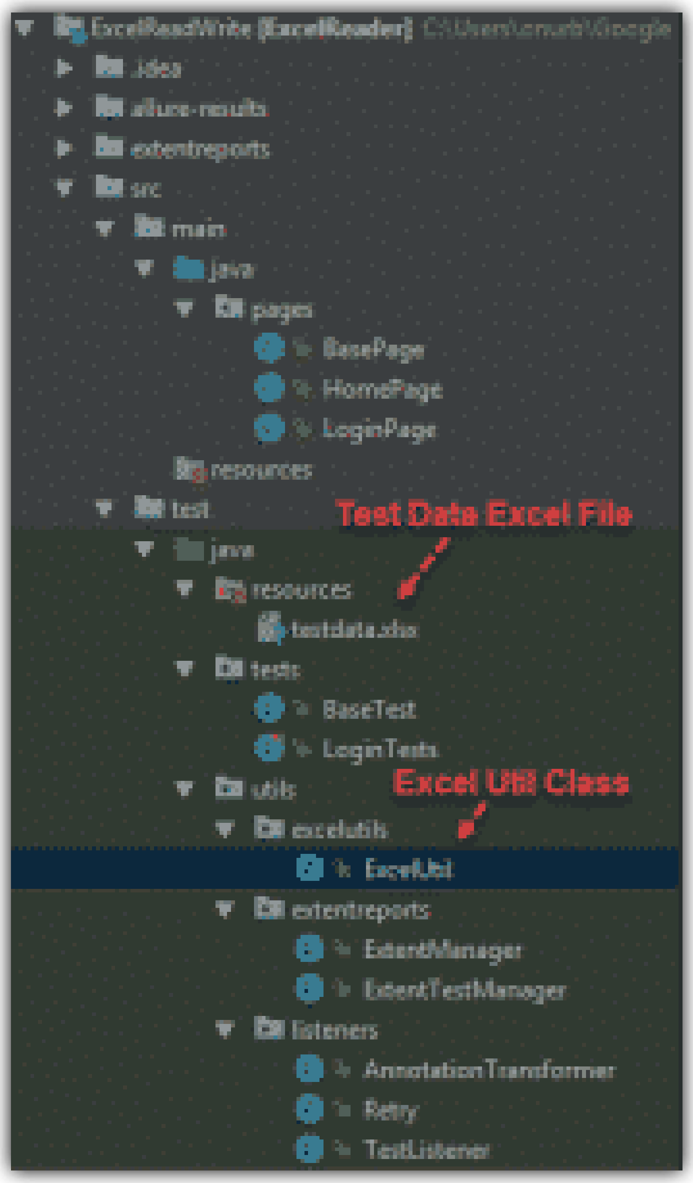
Data Driven Testing with Excel in Selenium (2019 Update)

Hi all, in this article I will describe you **how to use excel files in your test automation projects** for **data driven testing**. We can do data driven testing in several ways. We can use **TestNG data provider** for small data sets such as 3-4 different login data or if we have more data we can choose to use **excel files**or we can **store the test data in a database**. In this article, I will explain how to use excel files to store all test-related data. In order to manipulate excel files, I mean **read the excel file and write to an excel file**, we can use **Apache POI** API. I will show you how to integrate POI libraries into our test project.

I will go on with our **Allure reporting example**, it comprises of **Page Object Model (POM)** pattern, **ExtentReports Reporting**, and **Allure Reporting**features and we will add excel manipulation capability in that project. In order to do that, I will add an **ExcelUtil class** and this class does all kinds of excel operations. Here is the final snapshot of our project.



I will go step by step 😉 Don't worry! I hope, I will do by best, and you will get the topic without any problem. 😉

Step-1: Add Apache POI Dependencies

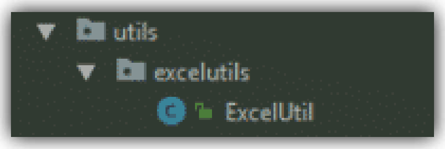
In order to use **Apache POI** libraries in your project, you should add required dependencies into your **pom.xml** as shown below.

```
<!-- https://mvnrepository.com/artifact/org.apache.p
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>4.0.1</version>
</dependency>

<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>4.0.1</version>
</dependency>
```

Step-2: Create an ExcelUtil Class for Data Driven Testing

In order to **manipulate excel files** and do **excel operations**, we should create an excel file and called it "**ExcelUtil**" under excelutils package as shown below.



In this file, I wrote all excel operation methods.

setExcelFileSheet: This method has two parameters: "**Test data excel file name**" and "**Excel sheet name**". It creates FileInputStream and set excel file and excel sheet to **excelWBook** and **excelWSheet** variables.

getCellData: This method **reads the test data from the Excel cell**. We are passing row number and column number as parameters.

getRowData: This method takes row number as a parameter and **returns the data of the given row number**.

setCellData: This method gets excel file, row, and column number and **set a value to that cell**.

and I have **setters** and **getters** for **rows** and **columns**. I will use all of the methods in test classes.

Here is the implementation of **ExcelUtil Class**:

ExcelUtil.java Java

```java
package utils.excelutils;

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;
import org.openqa.selenium.Platform;

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

import static tests.BaseTest.testDataExcelFileName

/**
 * Created by obaskirt on 28-Oct-17.
 * Updated by obaskirt on 02-Apr-2019
 */
public class ExcelUtil {
    //Main Directory of the project
    public static final String currentDir = System

    //Location of Test data excel file
    public static String testDataExcelPath = null;

    //Excel WorkBook
    private static XSSFWorkbook excelWBook;

    //Excel Sheet
    private static XSSFSheet excelWSheet;

    //Excel cell
    private static XSSFCell cell;

    //Excel row
    private static XSSFRow row;

    //Row Number
    public static int rowNumber;

    //Column Number
    public static int columnNumber;

    //Setter and Getters of row and columns
    public static void setRowNumber(int pRowNumber)
        rowNumber = pRowNumber;
    }

    public static int getRowNumber() {
        return rowNumber;
    }

    public static void setColumnNumber(int pColumnN
        columnNumber = pColumnNumber;
    }

    public static int getColumnNumber() {
        return columnNumber;
    }

    // This method has two parameters: "Test data e
    // It creates FileInputStream and set excel fil
    public static void setExcelFileSheet(String she
        //MAC or Windows Selection for excel path
        if (Platform.getCurrent().toString().equals
            testDataExcelPath = currentDir + "//src
        } else if (Platform.getCurrent().toString()
            testDataExcelPath = currentDir + "\\src
        }
        try {
            // Open the Excel file
            FileInputStream ExcelFile = new FileInp
            excelWBook = new XSSFWorkbook(ExcelFile
            excelWSheet = excelWBook.getSheet(sheet
        } catch (Exception e) {
            try {
                throw (e);
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        }
    }

    //This method reads the test data from the Exce
    //We are passing row number and column number a
    public static String getCellData(int RowNum, in
        try {
            cell = excelWSheet.getRow(RowNum).getCe
            DataFormatter formatter = new DataForma
            String cellData = formatter.formatCellV
            return cellData;
        } catch (Exception e) {
            throw (e);
        }
    }

    //This method takes row number as a parameter a
    public static XSSFRow getRowData(int RowNum) {
        try {
            row = excelWSheet.getRow(RowNum);
            return row;
        } catch (Exception e) {
            throw (e);
        }
    }

    //This method gets excel file, row and column n
    public static void setCellData(String value, in
        try {
            row = excelWSheet.getRow(RowNum);
            cell = row.getCell(ColNum);
            if (cell == null) {
                cell = row.createCell(ColNum);
                cell.setCellValue(value);
            } else {
                cell.setCellValue(value);
            }
            // Constant variables Test Data path an
            FileOutputStream fileOut = new FileOutp
            excelWBook.write(fileOut);
            fileOut.flush();
            fileOut.close();
        } catch (Exception e) {
            try {
                throw (e);
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        }
    }
}
```

Step-3: Set Data Excel File Name in BaseTest Class

I should also add **testDataExcelFileName** in BaseTest class because **all tests use the same excel file** but **their sheets are different**.

Step-4: Setup Test Data in Test Class

We need to **set the excel file and sheet name before starting the tests**. We have to do it in related test class because each test class has different test data and their sheets in the global test data excel are different too.

```
[illegible]
```

Step-5: Create a Test Excel File

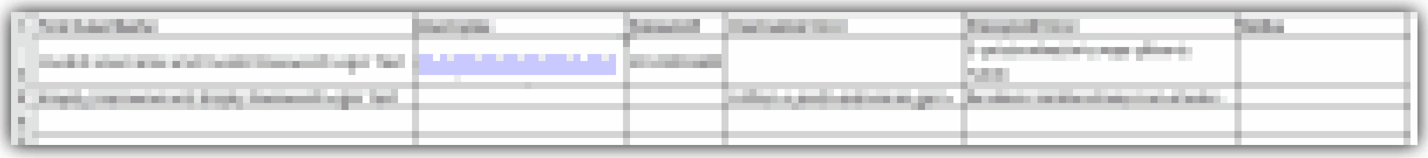
Now, it is time to construct our test excel data file for data-driven testing. In this example, I will modify our login scenarios (tests). **First one is "invalid username" and "invalid password" test**. I will store the following variables in the **LoginData sheet**:

- **username (invalid)**
- **password (invalid)**
- **username error message**
- **password error message**
- **test status (automation code will update after test execution.)**

For the **second test**, I will test **the empty username and empty password** case. Thus, my data will be like that:

- **username (empty)**
- **password (empty)**
- **username error message**
- **password error message**
- **test status (automation code will update after test execution.)**

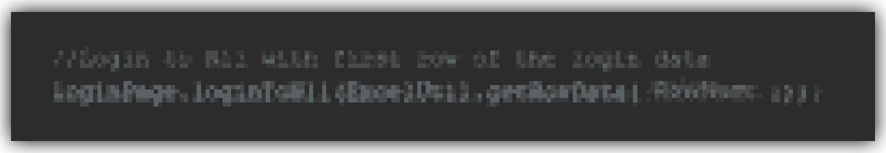
Here is what it looks like:



[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]		[illegible]	
[illegible]			[illegible]	[illegible]	

Step-6: Modify Test and Page Classes

In LoginTests class, we should start to modify our code for data-driven testing. First, let's start with "**invalidLoginTest_InvalidUserNameInvalidPassword**" test. I will use "getRowData" method for logintoN11 operation. In order to get first test data values (first row), we should use the below code:



```
//Login to N11 with first row of the login data
LoginPage.loginToN11(ExcelUtil.getRowData( RowNum: 1));
```