

## Data Driven Framework in Selenium Using Apache POI?

As we know that Nowadays Data Driven Framework is one of the most important Framework that we used in automation testing in any Web application and in which test data set is created in the excel sheet, and is then imported into our application during we perform automation testing.

We know that to test any functionality of an application, we need to test with multiple input values sometimes it counts is 100-1000. So it is a headache in performing a test on an application with 100-1000 data values, So Data Driven Framework allows users to separate their data from the code for re-usability purpose. and also it allows to create test scripts, where test data or input/output values are read from an external data files instead of using a hard code in each time when the test is run.

Below I mentioned all the External files that we used in performing Data Driven Testing with Apache POI in Selenium WebDriver.

- Excel files
- CSV files
- ODBC sources
- ADO objects

Here we are using excel sheet as an external data source. and our Test script is written in eclipse.

What is Apache POI?

Before Start accessing the data from excel files, we have to know about Apache POI and Why we are using POI in this Framework. So POI (Poor Obfuscation Implementation) is an API written in Java to support read and write operations – modifying office files. This is the most common API used for [Selenium data-driven tests](#).

What do we need to implement Data Driven Framework?

In order to follow this approach we must have Eclipse, TestNG properly configured.

If TestNG is not configured you can configure from [How to install TestNG in Eclipse](#).

**Once done, we will look at:**

- Various interfaces of Apache POI.
- Integration of Apache POI in the Eclipse.
- Read Data from the Excel Sheet.
- Write data to the Excel Sheet.
- Advantages of using Apache POI with Selenium.

Interface in POI

So one of the most Remarkable features of POI is that it supports both Read and write operation on both .xls and .xlsx files.

Below I mentioned some Interface of POI:

- **XSSFWorkbook:** Represents workbook in xlsx file.
- **HSSFWorkbook:** Represents workbook in xls file.
- **XSSFSheet:** Represents a sheet in XLSX file.
- **HSSFSheet:** Represents a sheet in XLS file.
- **XSSFRow:** Represents a row in a sheet of XLSX file.
- **HSSFRow:** Represents a row in a sheet of XLS file.
- **XSSFCell:** Represents a cell in a row of XLSX file.
- **HSSFCell:** Represents a cell in a row of XLS file.

Now Follow the below Steps for Data-driven Framework in selenium webDriver.

## Integration of Apache POI in the Eclipse.

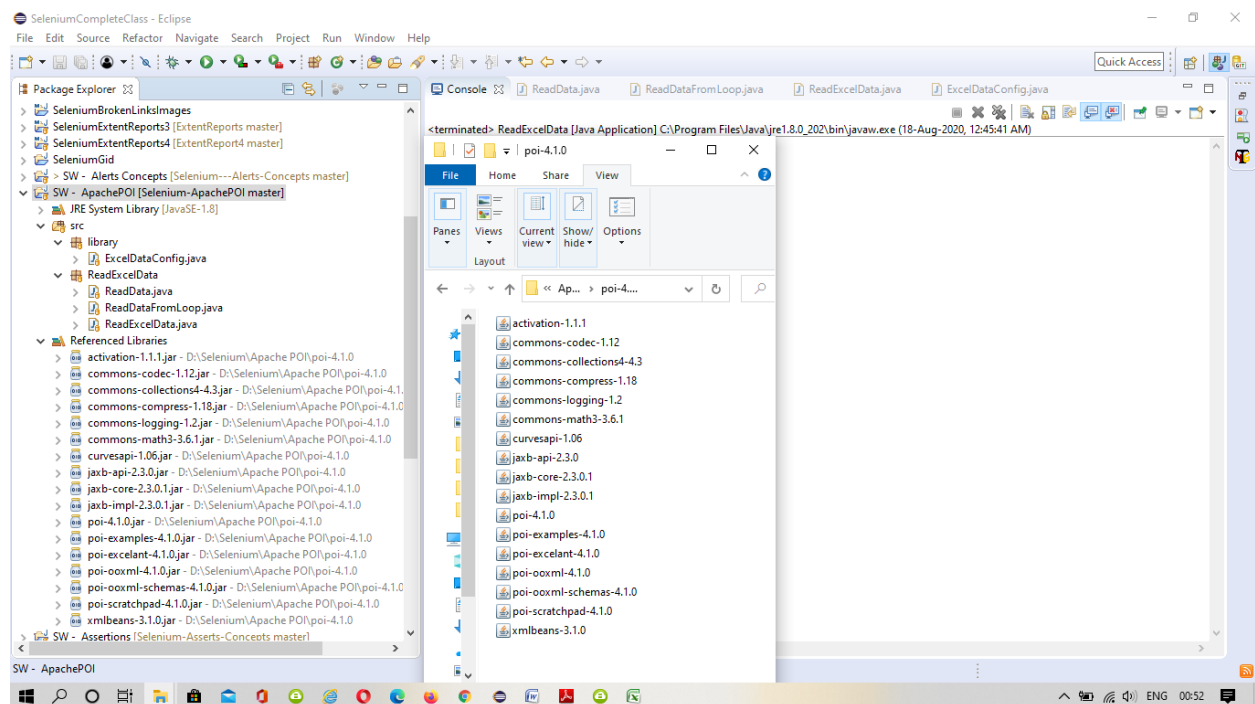
### Step #1)

Firstly, we need to configure Eclipse with **Apache POI**.

[Download](#) jar files for Apache POI.

### Step #2)

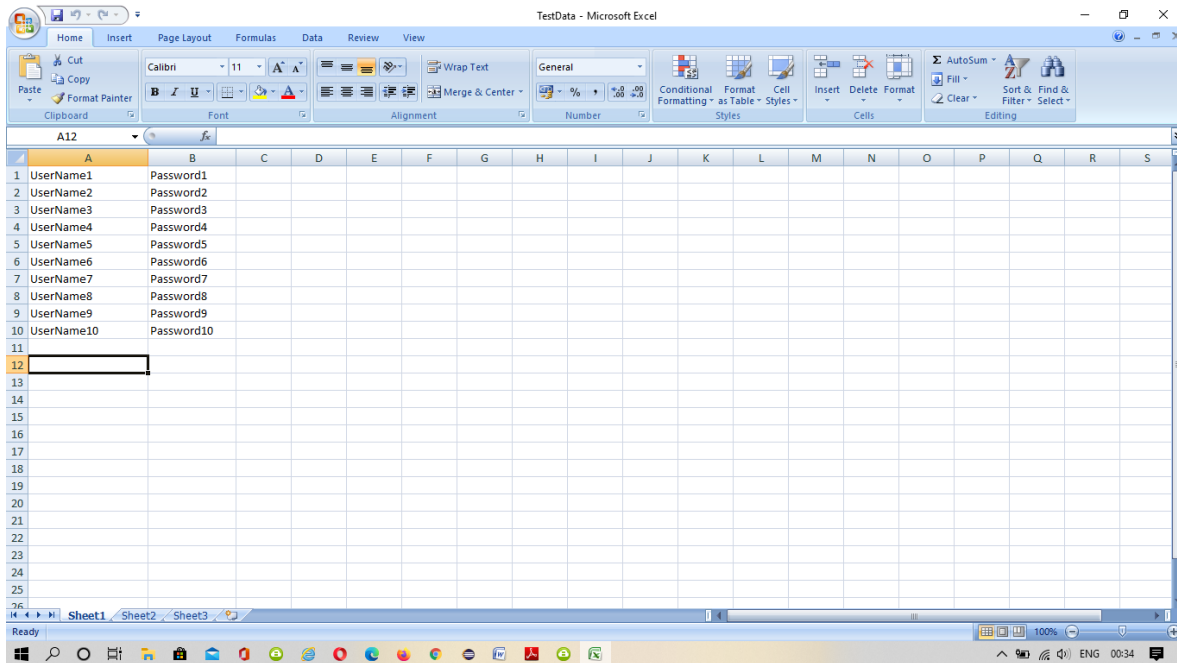
Unzip the jar file, and add the following jars to your project and configure them.



## Write Data in Excel-Sheet

### Step #3)

After configuring the respective jars, create an excel sheet enter some data in it and save it as filename.xls at your preferred location.



Here I used .xls file because I'm using HSSF SHEET in our code which Represents a sheet for .xls file.

## Read Data from Excel-Sheet

In my code file, I wrote all excel operation methods:

**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 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**.

**getAddress:** Gets the address of this cell

**getBooleanCellValue():** Get the value of the cell as a boolean.

**getColumnIndex():** Returns column index of this cell.

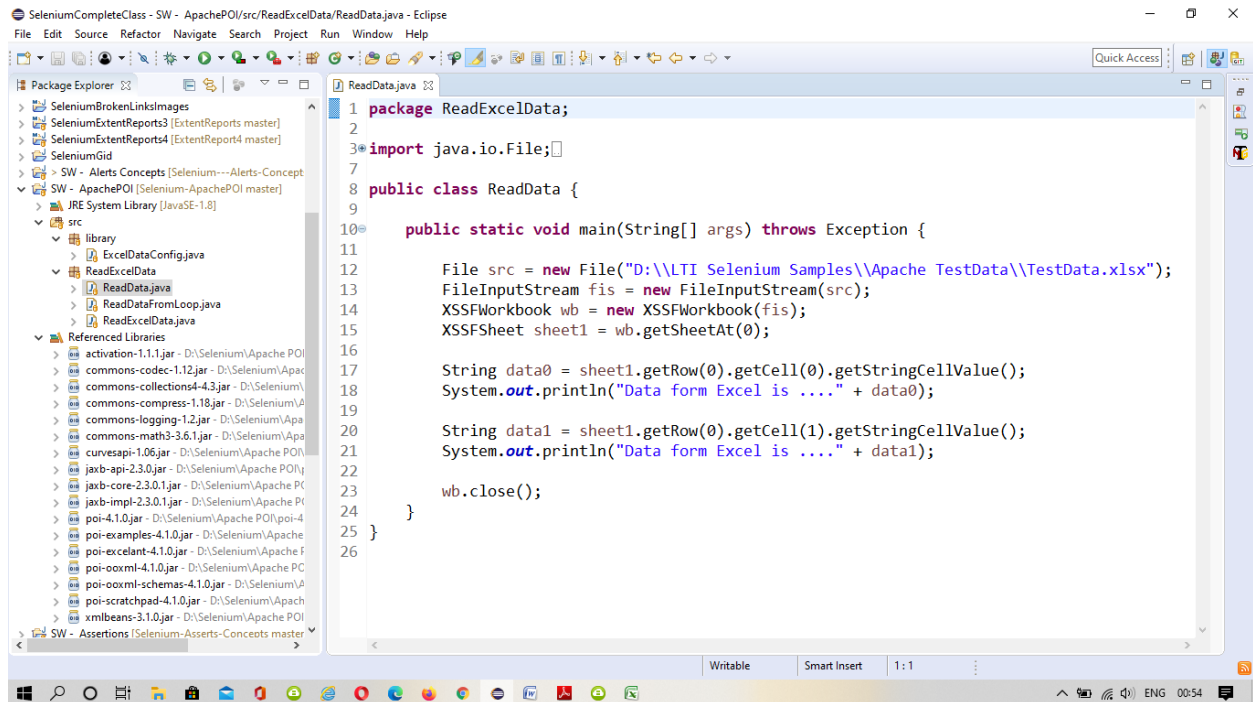
**getDateCellValue():** Get the value of the cell as a date.

**getNumericCellValue():** Get the value of the cell as a number.

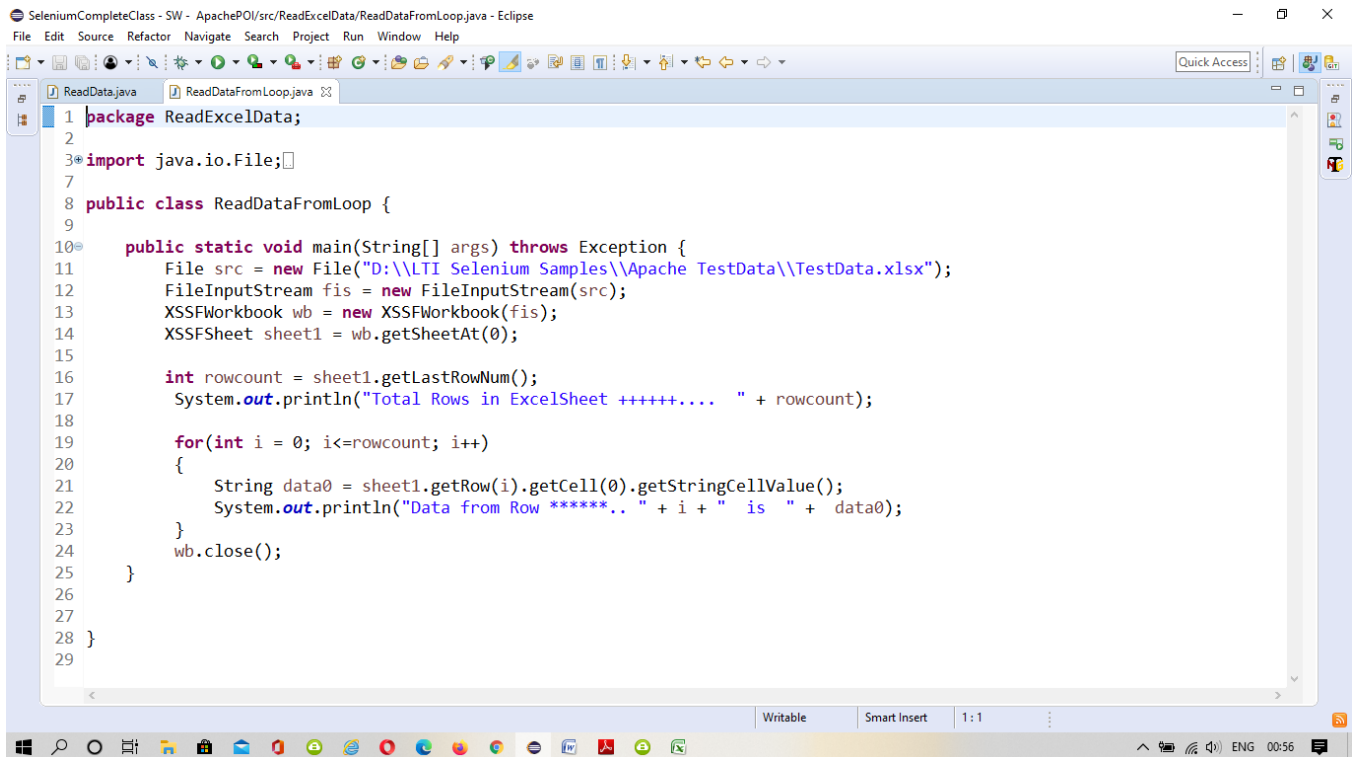
**getStringCellValue():** Get the value of the cell as a number.

#### Step #4)

Now let us follow the sample code to read data from the excel sheet and below i mention code for ReadData



```
1 package ReadExcelData;
2
3 import java.io.File;
4
5 public class ReadData {
6
7     public static void main(String[] args) throws Exception {
8
9         File src = new File("D:\\LTI Selenium Samples\\Apache TestData\\TestData.xlsx");
10        FileInputStream fis = new FileInputStream(src);
11        XSSFWorkbook wb = new XSSFWorkbook(fis);
12        XSSFSheet sheet1 = wb.getSheetAt(0);
13
14        String data0 = sheet1.getRow(0).getCell(0).getStringCellValue();
15        System.out.println("Data form Excel is ...." + data0);
16
17        String data1 = sheet1.getRow(0).getCell(1).getStringCellValue();
18        System.out.println("Data form Excel is ...." + data1);
19
20        wb.close();
21    }
22 }
```



```
1 package ReadExcelData;
2
3 import java.io.File;
4
5 public class ReadDataFromLoop {
6
7     public static void main(String[] args) throws Exception {
8         File src = new File("D:\\LTI Selenium Samples\\Apache TestData\\TestData.xlsx");
9         FileInputStream fis = new FileInputStream(src);
10        XSSFWorkbook wb = new XSSFWorkbook(fis);
11        XSSFSheet sheet1 = wb.getSheetAt(0);
12
13        int rowcount = sheet1.getLastRowNum();
14        System.out.println("Total Rows in ExcelSheet +++++.... " + rowcount);
15
16        for(int i = 0; i<=rowcount; i++)
17        {
18            String data0 = sheet1.getRow(i).getCell(0).getStringCellValue();
19            System.out.println("Data from Row *****. " + i + " is " + data0);
20        }
21        wb.close();
22    }
23 }
24
25
26
27
28
29
```

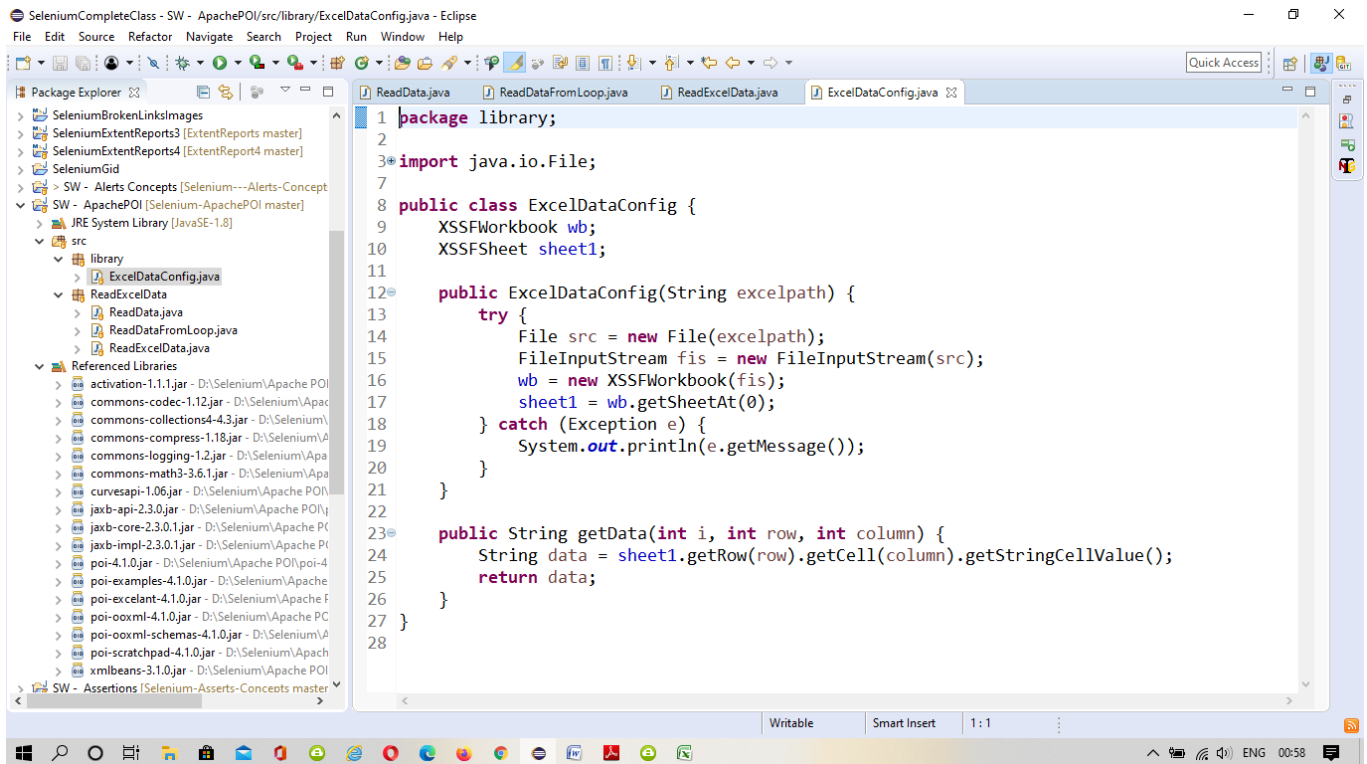
Here above I mentioned my all excel function and declare them in class ReadwriteExcel.

## Step #5)

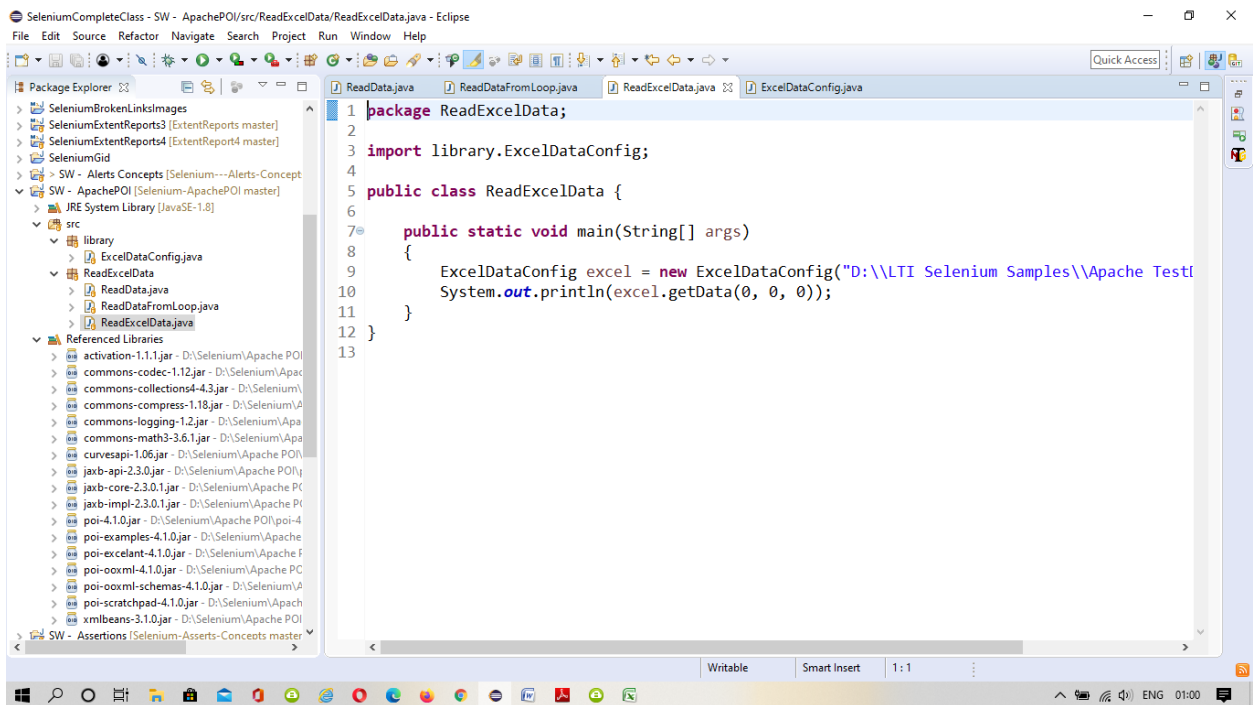
Right-click on the test case class and click on Run as → TestNG Test.

**Apache POI** imports data from the excel sheet and uses it to log into our application. Now that we saw how to read data from the excel sheet, let's look at Result cell in sheet.

Here for above code, we use HSSF for .xls files.



and when my above Test will be passed in my excel file i.e SampleXLSfile.xls Result will show :



**Note: If you encounter any problems during this process, please check the following points.**

- Make sure all the mentioned jars are added to the project and are properly configured.
- Required software is correctly installed.
- Proper use of an interface with respect to excel file, like HSSF for .xls and XSSF for .xlsx.
- Valid row and column index is used.
- Excel file must be closed before execution.
- Proper classes used for the excel file like XSSF used for .xlsx files and HSSF used for .xls files.

## **Advantages Of Using Data Driven Framework**

- Improves test coverage like input/output from and to a file is very crucial process it takes long hours. Apache POI plays a good Role and make this possible for Selenium Test Automation.
- Improves Re-usability of code.
- Less maintenance.
- Faster Execution.
- Permits better error handling.

Hope this blog helps you on performing Testing with Data Driven Framework in Selenium WebDriver Using Apache POI on your machine. Happy Installing TestNG.