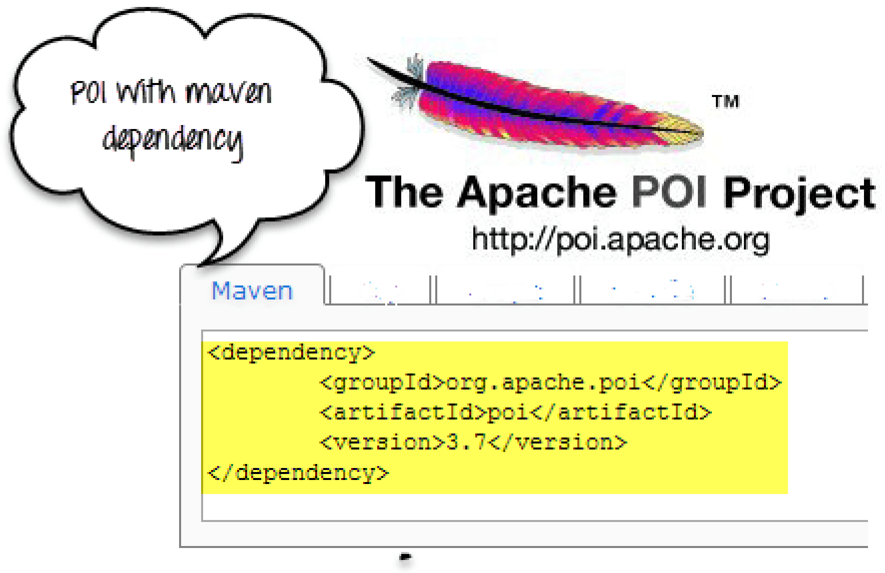
All About Excel in Selenium: POI & JXL

File IO is a critical part for any software process. We frequently create a file, open it & update something or delete it in our Computers. Same is the case with Selenium Automation. We need a process to manipulate files with Selenium.

Java provides us different classes for File Manipulation with Selenium. In this tutorial we are going to learn how can we read and write on Excel file with the help of Java IO package and Apache POI library.

Exporting Excel

- **How to handle excel file using POI (Maven POM Dependency)**



```
<dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi</artifactId>
        <version>3.7</version>
</dependency>
```

To read or write an Excel, Apache provide a very famous library POI. This library is capable enough to read and write both **XLS** and **XLSX** file format of excel.

To read **XLS** files an **HSSF** implementation is provided by POI library.

To read **XLSX** , **XSSF** implementation of **POI library** will be the choice. Let's study these implementations in detail.

If you are using maven in your project the maven dependency will be

<dependency>

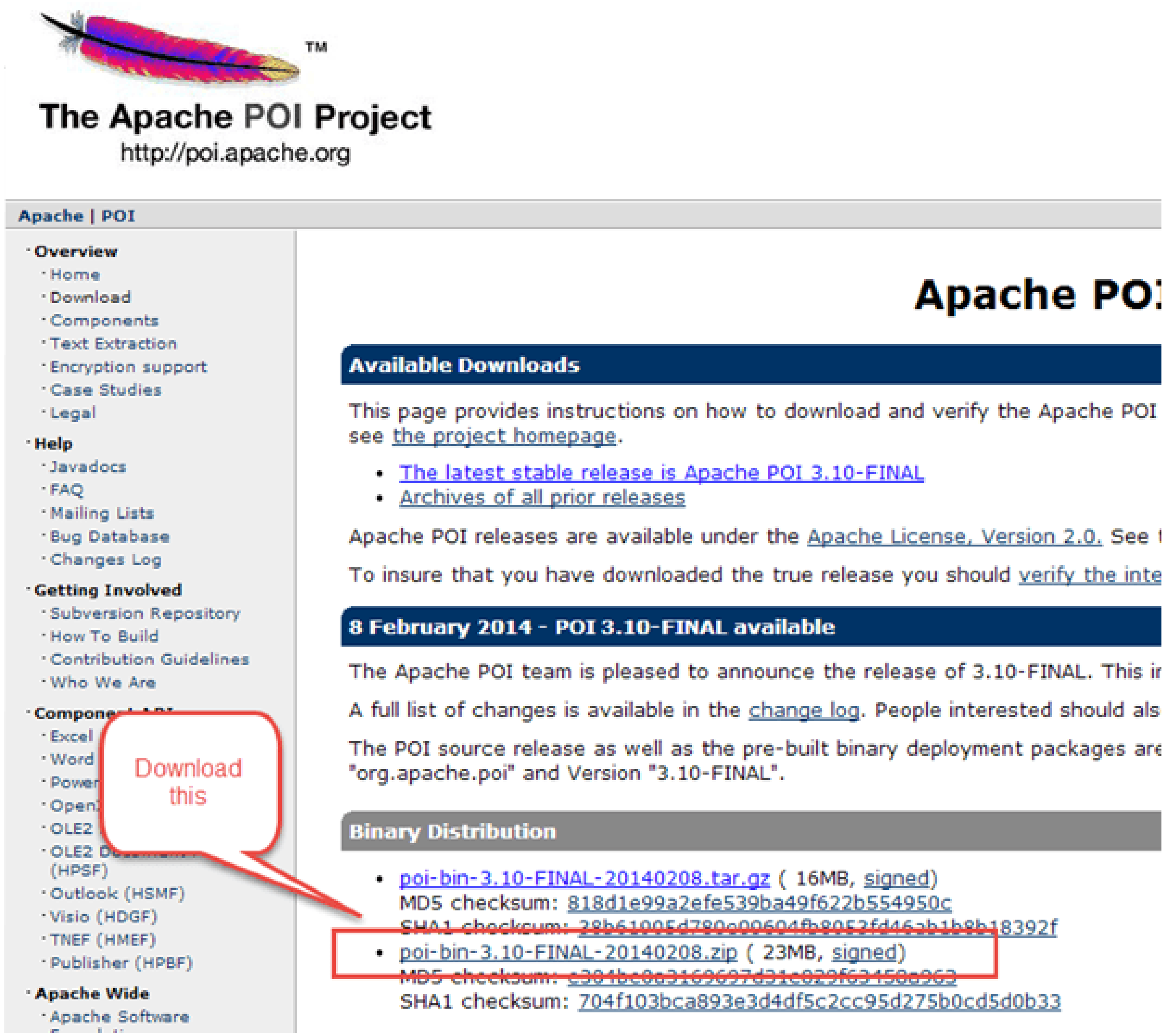
<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.9</version>

</dependency>

Or you can simply download the latest version POI jars from http://poi.apache.org/download.html & download poi-bin-3.10-FINAL-20140208.zip



Apache PO

Available Downloads

This page provides instructions on how to download and verify the Apache POI
see the project homepage.

- The latest stable release is Apache POI 3.10-FINAL
- Archives of all prior releases

Apache POI releases are available under the Apache License, Version 2.0. See

To insure that you have downloaded the true release you should verify the inte

8 February 2014 - POI 3.10-FINAL available

The Apache POI team is pleased to announce the release of 3.10-FINAL. This i

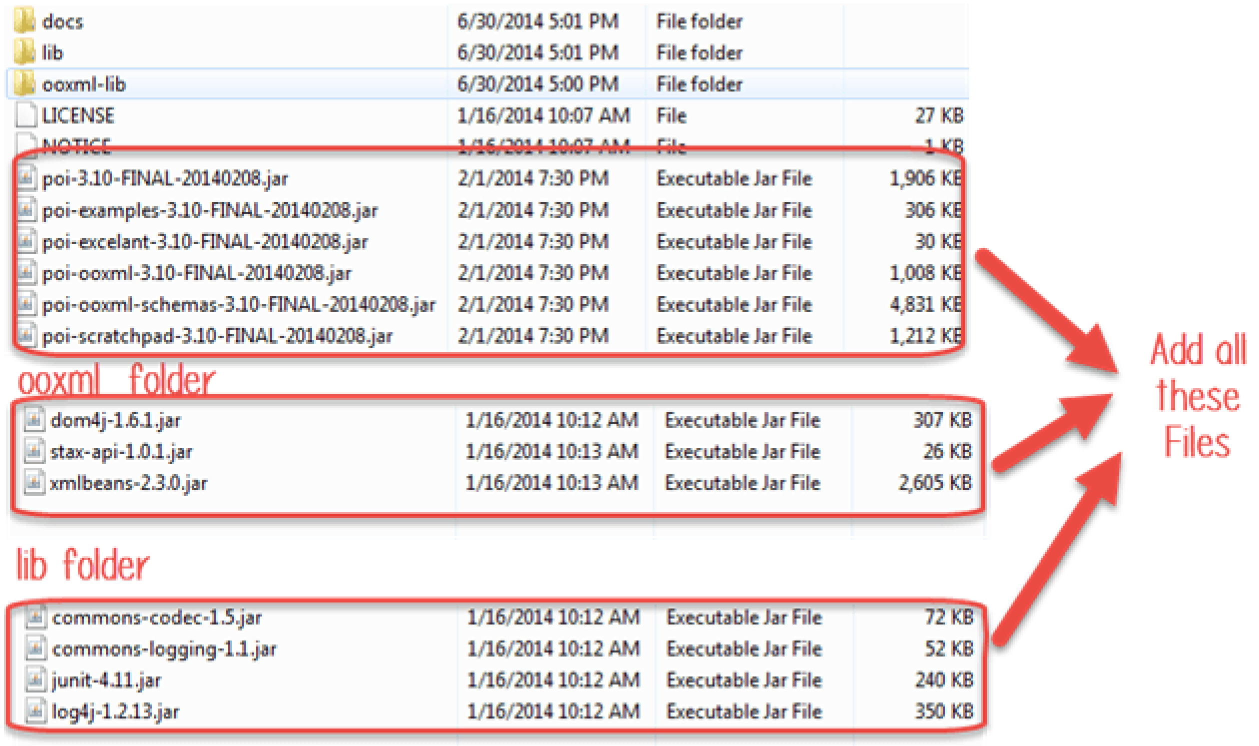
A full list of changes is available in the change log. People interested should als

The POI source release as well as the pre-built binary deployment packages are
"org.apache.poi" and Version "3.10-FINAL".

Binary Distribution

- poi-bin-3.10-FINAL-20140208.tar.gz (16MB, signed)
  MD5 checksum: 818d1e99a2efe539ba49f622b554950c
  SHA1 checksum: 28b61005d780e00604fb8053fd46ab1b8b18392f
- poi-bin-3.10-FINAL-20140208.zip (23MB, signed)
  MD5 checksum: c384bc0a3169697d31c829f63458a963
  SHA1 checksum: 704f103bca893e3d4df5c2cc95d275b0cd5d0b33

When you download the zip file for this jar , you need to unzip it and add these all jars to the class path of your project.

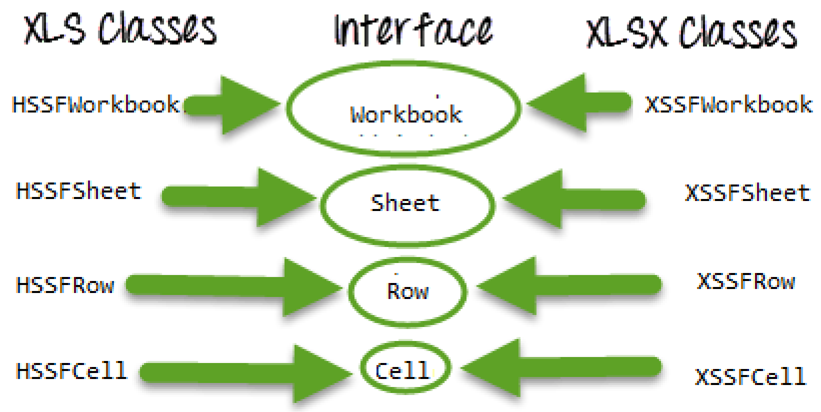


docs	6/30/2014 5:01 PM	File folder	
lib	6/30/2014 5:01 PM	File folder	
ooxml-lib	6/30/2014 5:00 PM	File folder	
LICENSE	1/16/2014 10:07 AM	File	27 KB
NOTICE	1/16/2014 10:07 AM	File	1 KB
poi-3.10-FINAL-20140208.jar	2/1/2014 7:30 PM	Executable Jar File	1,906 KB
poi-examples-3.10-FINAL-20140208.jar	2/1/2014 7:30 PM	Executable Jar File	306 KB
poi-excelant-3.10-FINAL-20140208.jar	2/1/2014 7:30 PM	Executable Jar File	30 KB
poi-ooxml-3.10-FINAL-20140208.jar	2/1/2014 7:30 PM	Executable Jar File	1,008 KB
poi-ooxml-schemas-3.10-FINAL-20140208.jar	2/1/2014 7:30 PM	Executable Jar File	4,831 KB
poi-scratchpad-3.10-FINAL-20140208.jar	2/1/2014 7:30 PM	Executable Jar File	1,212 KB

dom4j-1.6.1.jar	1/16/2014 10:12 AM	Executable Jar File	307 KB
stax-api-1.0.1.jar	1/16/2014 10:13 AM	Executable Jar File	26 KB
xmlbeans-2.3.0.jar	1/16/2014 10:13 AM	Executable Jar File	2,605 KB

commons-codec-1.5.jar	1/16/2014 10:12 AM	Executable Jar File	72 KB
commons-logging-1.1.jar	1/16/2014 10:12 AM	Executable Jar File	52 KB
junit-4.11.jar	1/16/2014 10:12 AM	Executable Jar File	240 KB
log4j-1.2.13.jar	1/16/2014 10:12 AM	Executable Jar File	350 KB

Classes and Interfaces in POI:



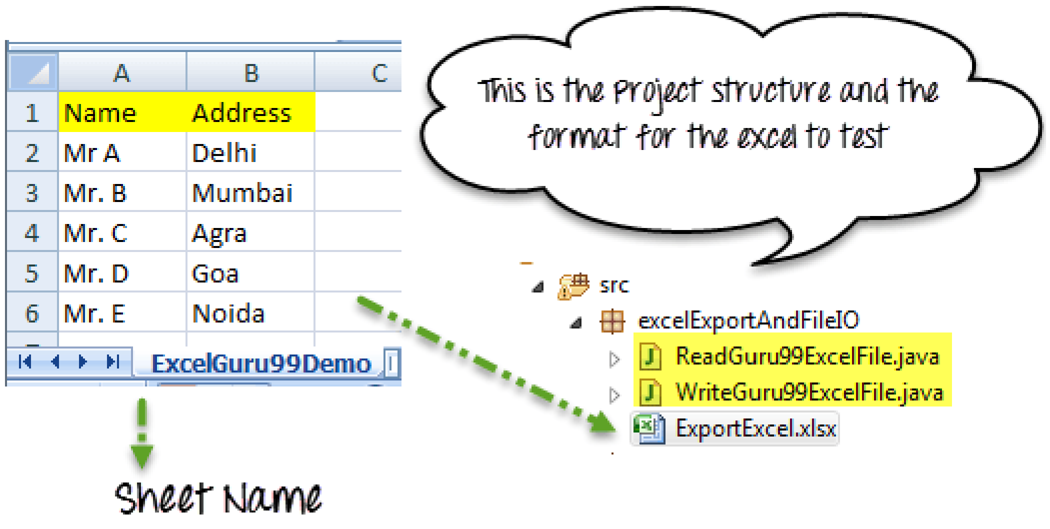
Following is a list of different Java Interfaces and classes in **POI** for reading **XLS** and **XLSX** file-

39) SSL Certificate
Selenium
40) XPath in Selen
Guide
42) Handling Ajax
Webdriver
43) Listeners and
WebDriver
49) Using Seleniu
44) Firefox Profile
WebDriver
50) How to use in
Webdriver
45) Breakpoints a
Selenium
46) Execute JavaS
using Selenium W
47) Using SoapUI
48) XSLT Report in
51) Selenium Inte
Answers
52) Flash Testing

- **Workbook**: XSSFWorkbook and HSSFWorkbook classes implement this interface.
- **XSSFWorkbook**: Is a class representation of XLSX file. ¶
- **HSSFWorkbook**: Is a class representation of XLS file.
- **Sheet**: XSSFSheet and HSSFSheet classes implement this interface.
- **XSSFSheet**: Is a class representing a sheet in a XLSX file.
- **HSSFSheet**: Is a class representing a sheet in a XLS file.
- **Row**: XSSFRow and HSSFRow classes implement this interface.
- **XSSFRow**: Is a class representing a row in sheet of XLSX file.
- **HSSFRow**: Is a class representing a row in sheet of XLS file.
- **Cell**: XSSFCell and HSSFCell classes implement this interface.
- **XSSFCell**: Is a class representing a cell in a row of XLSX file.
- **HSSFCell:** Is a class representing a cell in a row of XLS file.

Read/Write operation-

For our example we will consider below given excel file format



	A	B	C
1	Name	Address	
2	Mr A	Delhi	
3	Mr. B	Mumbai	
4	Mr. C	Agra	
5	Mr. D	Goa	
6	Mr. E	Noida	

ExcelGuru99Demo

```
src
  excelExportAndFileIO
    ReadGuru99ExcelFile.java
    WriteGuru99ExcelFile.java
  ExportExcel.xlsx
```

This is the Project structure and the format for the excel to test
Sheet Name

Read data from Excel file

Complete Example: Here we are trying to read data from excel file

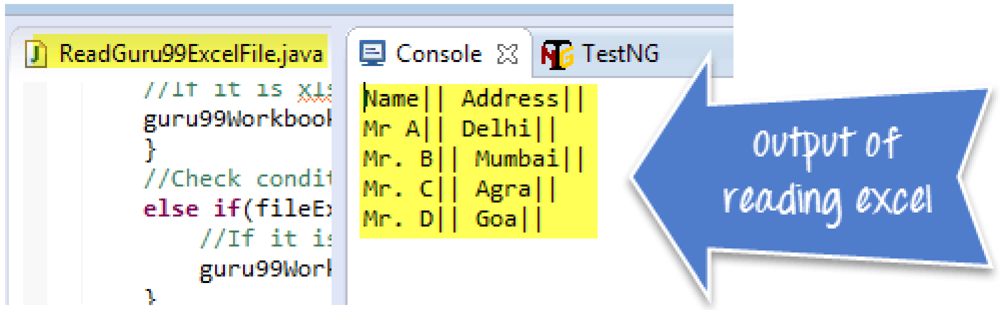
```
package excelExportAndFileIO;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

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.xssf.usermodel.XSSFWorkbook;

public class ReadGuru99ExcelFile {



    public void readExcel(String filePath,String fileName,String sheetName) throws IOException{

    //Create a object of File class to open xlsx file

    File file =    new File(filePath+"\\"+fileName);

    //Create an object of FileInputStream class to read excel file
```

```
FileInputStream inputStream = new FileInputStream(file);

Workbook guru99Workbook = null;

//Find the file extension by spliting file name in substring and getting only extension name

String fileExtensionName = fileName.substring(fileName.indexOf("."));

//Check condition if the file is xlsx file

if(fileExtensionName.equals(".xlsx")){

//If it is xlsx file then create object of XSSFWorkbook class

guru99Workbook = new XSSFWorkbook(inputStream);

}

//Check condition if the file is xls file

else if(fileExtensionName.equals(".xls")){

    //If it is xls file then create object of XSSFWorkbook class

    guru99Workbook = new HSSFWorkbook(inputStream);

}

//Read sheet inside the workbook by its name

Sheet guru99Sheet = guru99Workbook.getSheet(sheetName);

//Find number of rows in excel file

int rowCount = guru99Sheet.getLastRowNum()-guru99Sheet.getFirstRowNum();
```

```
//Create a loop over all the rows of excel file to read it

for (int i = 0; i < rowCount+1; i++) {

    Row row = guru99Sheet.getRow(i);

    //Create a loop to print cell values in a row

    for (int j = 0; j < row.getLastCellNum(); j++) {

        //Print excel data in console

        System.out.print(row.getCell(j).getStringCellValue()+"|| ");

    }

    System.out.println();

}



}



//Main function is calling readExcel function to read data from excel file

public static void main(String...strings) throws IOException{

//Create a object of ReadGuru99ExcelFile class

ReadGuru99ExcelFile objExcelFile = new ReadGuru99ExcelFile();
```

```
    //Prepare the path of excel file

    String filePath = System.getProperty("user.dir")+"\\src\\excelExportAndFileIO";

    //Call read file method of the class to read data

    objExcelFile.readExcel(filePath,"ExportExcel.xlsx","ExcelGuru99Demo");

    }

}
```

Note: We are not using the TestNG framework here. Run the class as Java Application



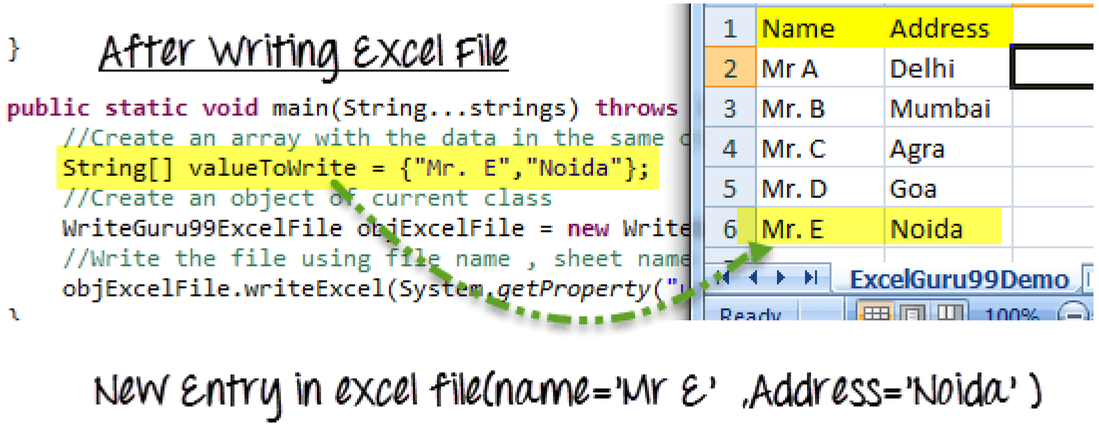
```
Name|| Address||
Mr A|| Delhi||
Mr. B|| Mumbai||
Mr. C|| Agra||
Mr. D|| Goa||
```

- **Write data on Excel file**

Complete Example: Here we are trying to write data from excel file by adding new row in excel file

```
package excelExportAndFileIO;

import java.io.File;

import java.io.FileInputStream;

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

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

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.xssf.usermodel.XSSFWorkbook;

public class WriteGuru99ExcelFile {



    public void writeExcel(String filePath,String fileName,String sheetName,String[] dataToWrite) throws IOExceptio
n{

        //Create a object of File class to open xlsx file

        File file =    new File(filePath+"\\"+fileName);

        //Create an object of FileInputStream class to read excel file

        FileInputStream inputStream = new FileInputStream(file);

        Workbook guru99Workbook = null;

        //Find the file extension by spliting file name in substing and getting only extension name

        String fileExtensionName = fileName.substring(fileName.indexOf("."));
    //Check condition if the file is xlsx file

    if(fileExtensionName.equals(".xlsx")){

    //If it is xlsx file then create object of XSSFWorkbook class

    guru99Workbook = new XSSFWorkbook(inputStream);

    }

    //Check condition if the file is xls file

    else if(fileExtensionName.equals(".xls")){

        //If it is xls file then create object of XSSFWorkbook class

        guru99Workbook = new HSSFWorkbook(inputStream);

    }



//Read excel sheet by sheet name

Sheet sheet = guru99Workbook.getSheet(sheetName);

//Get the current count of rows in excel file

int rowCount = sheet.getLastRowNum()-sheet.getFirstRowNum();

//Get the first row from the sheet

Row row = sheet.getRow(0);
```

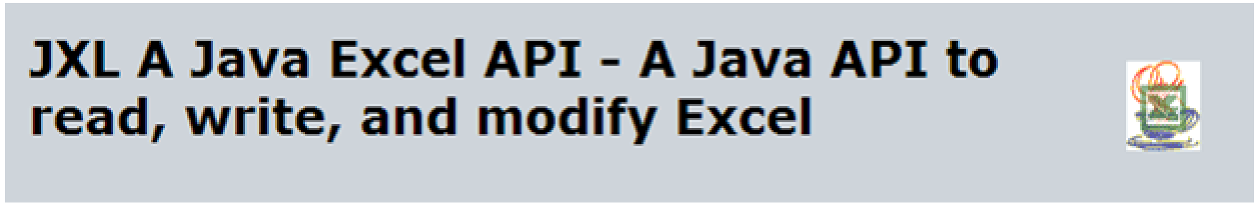
```
    //Create a new row and append it at last of sheet

    Row newRow = sheet.createRow(rowCount+1);

    //Create a loop over the cell of newly created Row

    for(int j = 0; j < row.getLastCellNum(); j++){

        //Fill data in row

        Cell cell = newRow.createCell(j);

        cell.setCellValue(dataToWrite[j]);

    }

    //Close input stream

    inputStream.close();

    //Create an object of FileOutputStream class to create write data in excel file

    FileOutputStream outputStream = new FileOutputStream(file);

    //write data in the excel file

    guru99Workbook.write(outputStream);

    //close output stream

    outputStream.close();



    }
    public static void main(String...strings) throws IOException{

        //Create an array with the data in the same order in which you expect to be filled in excel file

        String[] valueToWrite = {"Mr. E","Noida"};

        //Create an object of current class

        WriteGuru99ExcelFile objExcelFile = new WriteGuru99ExcelFile();

        //Write the file using file name , sheet name and the data to be filled

        objExcelFile.writeExcel(System.getProperty("user.dir")+"\\src\\excelExportAndFileIO","ExportExcel.xlsx","Ex
celGuru99Demo",valueToWrite);

    }

}
```



```
}
public static void main(String...strings) throws
    //Create an array with the data in the same
    String[] valueToWrite = {"Mr. E","Noida"};
    //Create an object of current class
    WriteGuru99ExcelFile objExcelFile = new Write
    //Write the file using file name , sheet name
    objExcelFile.writeExcel(System.getProperty("
}
```

1	Name	Address
2	Mr A	Delhi
3	Mr. B	Mumbai
4	Mr. C	Agra
5	Mr. D	Goa
6	Mr. E	Noida

Excel Manipulation using JXL API



JXL A Java Excel API - A Java API to read, write, and modify Excel

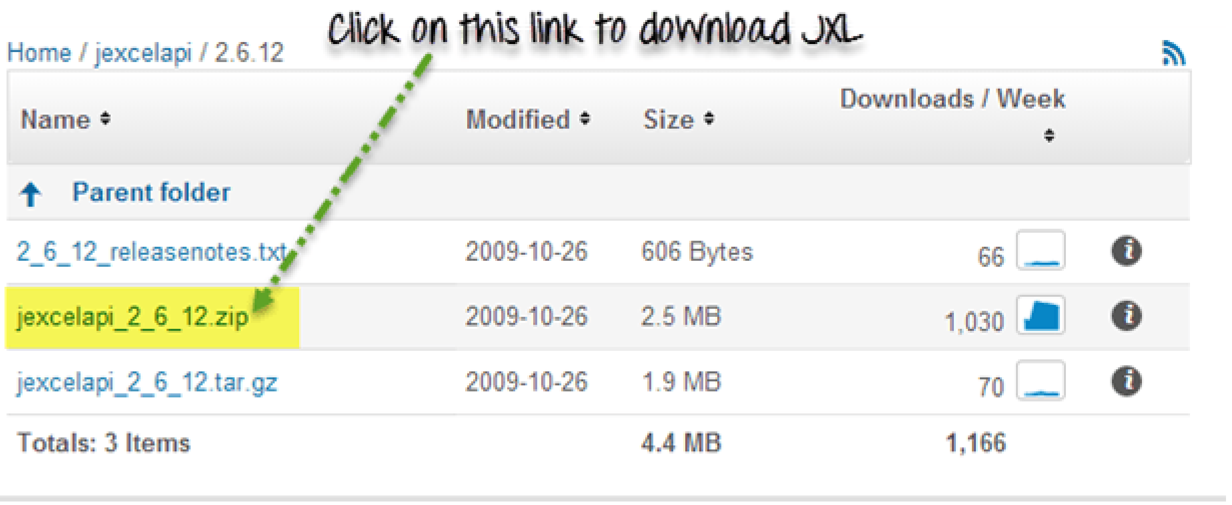
JXL is also another famous jar for reading writing Excel files. Now a day's POI is used in most of the projects but before POI, JXL was only Java API for excel manipulation. It is a very small and simple API.

TIPS: *My suggestion is not to use JXL in any new project because the library is not in active development from 2010 and lack in feature in compare to POI API.*

Download JXL:

If you want to work with JXL you can download it from this link

http://sourceforge.net/projects/jexcelapi/files/jexcelapi/2.6.12/



Home / jexcelapi / 2.6.12

Name ♦	Modified ♦	Size ♦	Downloads / Week ♦
Parent folder			
2_6_12_releasenotes.txt	2009-10-26	606 Bytes	66
jexcelapi_2_6_12.zip	2009-10-26	2.5 MB	1,030
jexcelapi_2_6_12.tar.gz	2009-10-26	1.9 MB	70
Totals: 3 Items		4.4 MB	1,166

You can also get demo example inside this zipped file for JXL.

Some of the features:

- JXL is able to read Excel 95, 97, 2000,XP , 2003 workbook.
- We can work with English, French, Spanish, German.
- Copying a Chart and image insertion in excel is possible

Drawback:

- We can write excel 97 and later only (writing in Excel 95 is not supported).
- JXL does not support XLSX format of excel file.
- It Generate spreadsheet in Excel 2000 format.

Summary:

- Excel file can be read by Java IO operation. For that, we need to use **Apache POI Jar**.
- There are two kind of workbook in excel file, **XLSX** and **XLS** files.
- POI has different Interfaces Workbook, Sheet, Row, Cell .
- These interfaces are implemented by corresponding **XLS**(**HSSFWorkbook, HSSFSheet, HSSFRow, HSSFCell**) and **XLSX**(**XSSFWorkbook, XSSFSheet, XSSFRow, XSSFCell**) file manipulation classes.
- JXL is another API for excel manipulation.
- JXL cannot work with XLSX format of excel.

‹ Prev Next›

RELATED ARTICLES