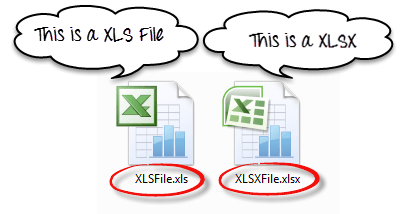
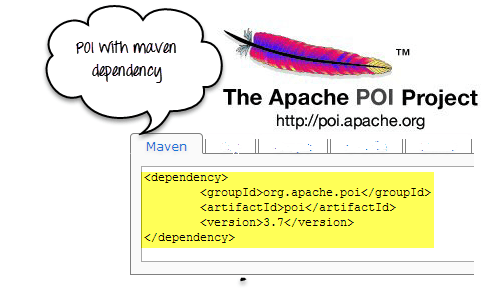
File IO is a critical part of any software process. We frequently create a file, open it & update something or delete it in our Computers. Same is the case with Selenium Automation. We need a process to manipulate files with Selenium.

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](http://www.guru99.com/excel-tutorials.html) file with the help of [Java](http://www.guru99.com/java-tutorial.html) IO package and [Apache](http://www.guru99.com/apache.html) POI library.

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc1.png)

**Exporting Excel**

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

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc2.png)

To read or write an Excel,Apache provides 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>**

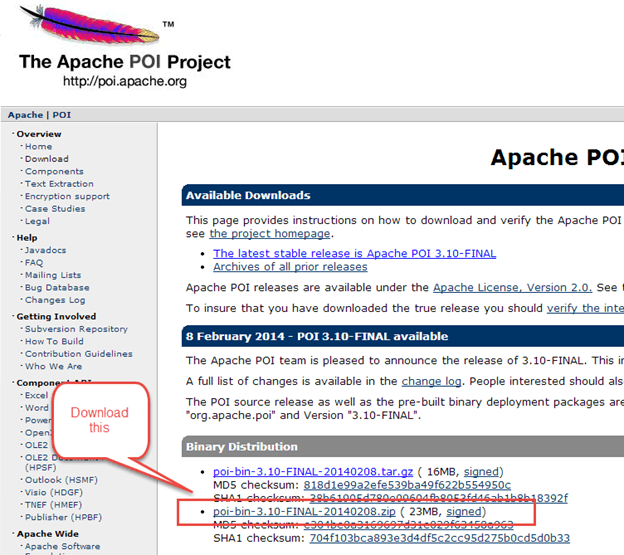
**<groupId>org.apache.poi</groupId>**

**<artifactId>poi</artifactId>**

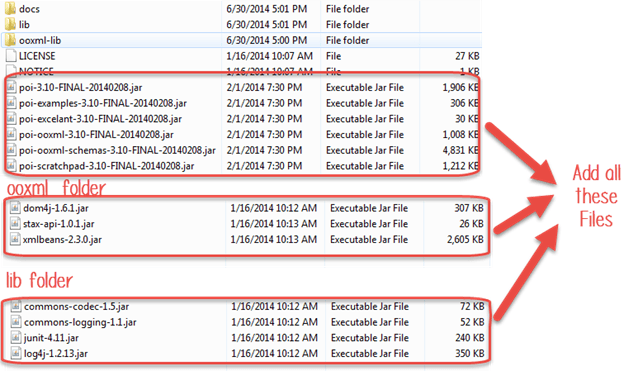
**<version>3.9</version>**

**</dependency>**

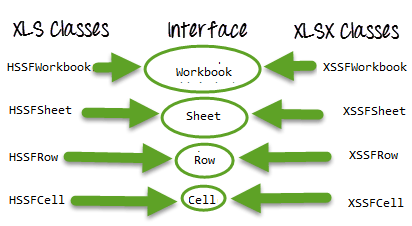
Or you can simply download the latest version POI jars from <http://poi.apache.org/download.html> & download [poi-bin-3.10-FINAL-20140208.zip](http://www.apache.org/dyn/closer.cgi/poi/release/bin/poi-bin-3.10-FINAL-20140208.zip)

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc3.png)

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.

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc4.png)

**Classes and Interfaces in POI:**

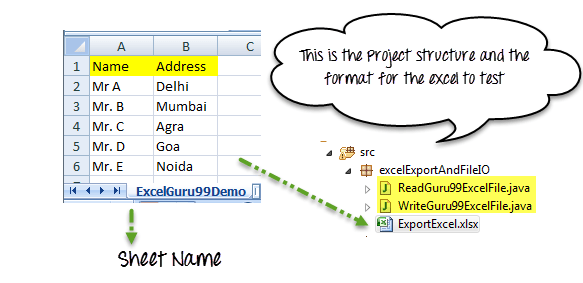
[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc5.png)

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

* **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 an XLSX file.
* **HSSFSheet**: Is a class representing a sheet in an XLS file.
* **Row**: XSSFRow and HSSFRow classes implement this interface.
* **XSSFRow**: Is a class representing a row in the sheet of XLSX file.
* **HSSFRow**: Is a class representing a row in the 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

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc6.png)

**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 an 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 splitting 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 an object of ReadGuru99ExcelFile class

ReadGuru99ExcelFile objExcelFile = new ReadGuru99ExcelFile();

//Prepare the path of excel file

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

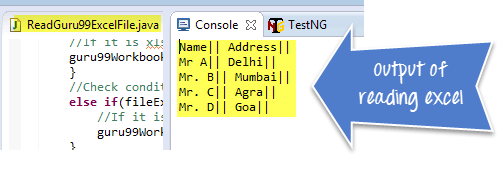
//Call read file method of the class to read data

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

}

}

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

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc7.png)

**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 IOException{

//Create an 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 splitting 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 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

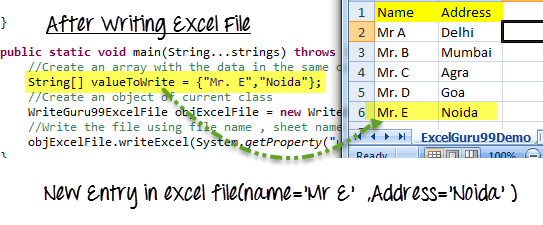
WriteGuru99ExcelFile objExcelFile = new WriteGuru99ExcelFile();

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

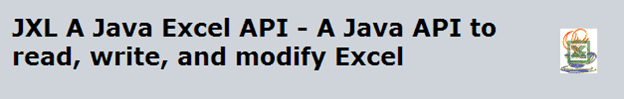
objExcelFile.writeExcel(System.getProperty("user.dir")+"\\src\\excelExportAndFileIO","ExportExcel.xlsx","ExcelGuru99Demo",valueToWrite);

}

}

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc8.png)

**Excel Manipulation using JXL API**

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc9.png)

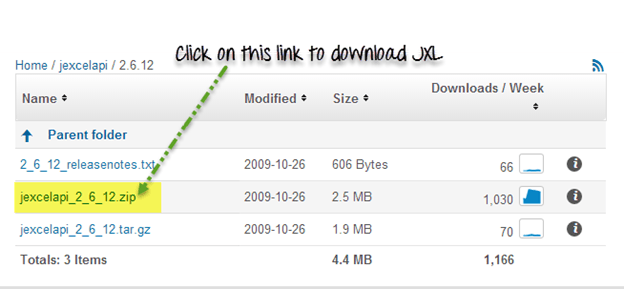
JXL is also another famous jar for reading writing Excel files. Now a day's POI is used in most of the projects, but before POI, JXL was only Java API for Excel manipulation. It is a very small and simple API.

TIPS: *My suggestion is not to use JXL in any new project because the library is not in active development from 2010 and lack of the 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/>

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc10.png)

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 Generates 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 kinds of a 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.