



Home > Java

# Read / Write Excel File In Java Using Apache POI

By [Viral Patel](#) on November 28, 2012



Apache POI is a powerful Java library to work with different Microsoft Office file formats such as Excel, Power point, Visio, MS Word etc. The name POI was originally an acronym for **Poor Obfuscation Implementation**, referring humorously to the fact that the file formats seemed to be deliberately obfuscated, but poorly, since they were successfully reverse-engineered.

In this tutorial we will use Apache POI library to perform different functions on Microsoft Excel spreadsheet.

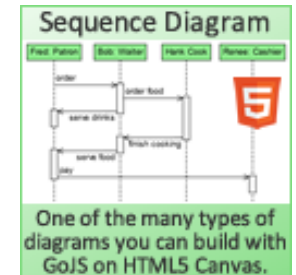
Let's get started.

## Tools & Technologies:

1. Java JDK 1.5 or above
2. Apache POI library v3.8 or above ([download](#))
3. Eclipse 3.2 above (optional)

## 1. Add Apache POI dependency

Make sure to include apache poi jar file to your project. If your project uses Maven as dependency management, add following in your Pom.xml file.



Subscribe

 Get our Articles via Email. Enter your email.

Your E-Mail

[Sign up](#)

```
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi</artifactId>
  <version>3.8</version>
</dependency>
```

If you are not using Maven then you can directly add required JAR files in your classpath.

1. Download [poi-2.5.1.jar](#) (or in this case 3.8) jar file.
2. Include this file in your projects class path.
3. Create new java project in eclipse with auto generated main function.

## 2. Read Excel File

To read an excel file, Apache POI provides certain easy-to-use APIs. In below sample code we use different classes from POI library to read content of cell from excel file. This is for quick reference.

```
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
//...
FileInputStream file = new FileInputStream(new File("C:\\test.xls"));

//Get the workbook instance for XLS file
HSSFWorkbook workbook = new HSSFWorkbook(file);

//Get first sheet from the workbook
HSSFSheet sheet = workbook.getSheetAt(0);

//Get iterator to all the rows in current sheet
Iterator<Row> rowIterator = sheet.iterator();

//Get iterator to all cells of current row
Iterator<Cell> cellIterator = row.cellIterator();
```

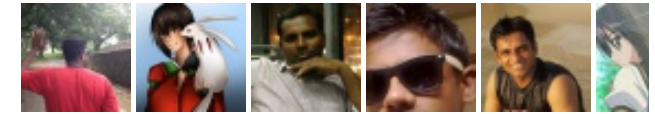
Notice how each class in POI library starts with **HSSF** prefix! e.g. HSSFWorkbook, HSSFSheet etc. HSSF stands for **Horrible Spreadsheet Format!** I'm not kidding.. It really is.



ViralPatel.net



3,031 people like ViralPatel.net.



Facebook social plugin



Follow on Twitter @viralpatelnet

### Latest Posts

1. [AngularJS Controller Tutorial with Example](#)
2. [AngularJS: Introduction and Hello World example](#)
3. [How to access Static Fields in Freemarker FTL](#)
4. [Lazy Load Image & Wordpress Gravatar using JavaScript / JQuery](#)
5. [21 JavaScript Tips and Tricks for JavaScript Developers](#)
6. [CSS Styling Radio Button and Checkboxes](#)
7. [Java XPath Tutorial: How to Parse XML File using XPath in Java](#)
8. [Android Internet Connection Status & Network Change Receiver example](#)
9. [Android: Activity name must be specified error](#)
10. [Spring MVC Flash Attribute tutorial with example](#)

Similar to HSSF, POI has different prefix for other file formats too:

1. HSSF (Horrible SpreadSheet Format) – reads and writes Microsoft Excel (XLS) format files.
2. XSSF (XML SpreadSheet Format) – reads and writes Office Open XML (XLSX) format files.
3. HPSF (Horrible Property Set Format) – reads “Document Summary” information from Microsoft Office files.
4. HWPf (Horrible Word Processor Format) – aims to read and write Microsoft Word 97 (DOC) format files.
5. HSLF (Horrible Slide Layout Format) – a pure Java implementation for Microsoft PowerPoint files.
6. HDGF (Horrible DiaGram Format) – an initial pure Java implementation for Microsoft Visio binary files.
7. HPBF (Horrible PuBlisher Format) – a pure Java implementation for Microsoft Publisher files.
8. HSMF (Horrible Stupid Mail Format) – a pure Java implementation for Microsoft Outlook MSG files
9. DDF (Dreadful Drawing Format) – a package for decoding the Microsoft Office Drawing format.

## Working with .xlsx files

The classes we used in above code snippet, `HSSFWorkbook` and `HSSFSheet` works for `.xls` format. In order to work with newer xls format viz `.xlsx`, you need to see newer POI classes like:

```
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFSheet;
//...
FileInputStream file = new FileInputStream(new File("C:\\test.xlsx"));

//Get the workbook instance for XLS file
XSSFWorkbook workbook = new XSSFWorkbook (file);

//Get first sheet from the workbook
XSSFSheet sheet = workbook.getSheetAt(0);

//Get iterator to all the rows in current sheet
Iterator<Row> rowIterator = sheet.iterator();

//Get iterator to all cells of current row
Iterator<Cell> cellIterator = row.cellIterator();
```

Use `XSSFWorkbook` and `XSSFSheet` class in all of the below examples in order to make them work

open in browser PRO version Are you a developer? Try out the [HTML to PDF API](#)

Sponsors

Links



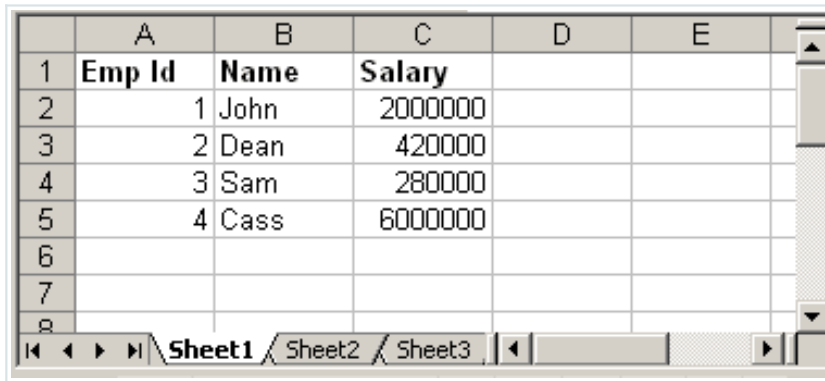
→ [Taj OS](#)



with .xlsx files.

Consider a sample excel file:

*test.xls*



The screenshot shows an Excel spreadsheet with a grid of columns A through E and rows 1 through 8. The first row (row 1) contains headers: 'Emp Id' in column A, 'Name' in column B, and 'Salary' in column C. The subsequent rows contain data: row 2 has '1' in A and 'John' in B; row 3 has '2' in A and 'Dean' in B; row 4 has '3' in A and 'Sam' in B; row 5 has '4' in A and 'Cass' in B. The 'Salary' column (C) is empty for all rows. The spreadsheet is titled 'Sheet1' and has tabs for 'Sheet1', 'Sheet2', and 'Sheet3' at the bottom.

	A	B	C	D	E
1	Emp Id	Name	Salary		
2	1	John	2000000		
3	2	Dean	420000		
4	3	Sam	280000		
5	4	Cass	6000000		
6					
7					
8					

We will read above xls file using Apache POI and prints the data.

```
try {  
  
    FileInputStream file = new FileInputStream(new File("C:\\test.xls"));  
  
    //Get the workbook instance for XLS file  
    HSSFWorkbook workbook = new HSSFWorkbook(file);  
  
    //Get first sheet from the workbook  
    HSSFSheet sheet = workbook.getSheetAt(0);  
  
    //Iterate through each rows from first sheet  
    Iterator<Row> rowIterator = sheet.iterator();  
    while(rowIterator.hasNext()) {  
        Row row = rowIterator.next();  
  
        //For each row, iterate through each columns  
        Iterator<Cell> cellIterator = row.cellIterator();  
        while(cellIterator.hasNext()) {  
  
            Cell cell = cellIterator.next();  
        }  
    }  
}
```

Learn PHP  
from the  
experts

35%  
OFF

PHP I:  
Foundations

Expires August 31

Buy Now >

```
        switch(cell.getCellType()) {
            case Cell.CELL_TYPE_BOOLEAN:
                System.out.print(cell.getBooleanCellValue() + "\t\t");
                break;
            case Cell.CELL_TYPE_NUMERIC:
                System.out.print(cell.getNumericCellValue() + "\t\t");
                break;
            case Cell.CELL_TYPE_STRING:
                System.out.print(cell.getStringCellValue() + "\t\t");
                break;
        }
    }
    System.out.println("");
}
file.close();
FileOutputStream out =
    new FileOutputStream(new File("C:\\test.xls"));
workbook.write(out);
out.close();

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

The above code is self explanatory. It read the sheet from workbook and iterate through each row and cell to print its values. Just note how we use different methods like `getBooleanCellValue`, `getNumericCellValue` etc to read cell value. Before reading a cell content, we need to first determine its type using method `cell.getCellType()` and then call appropriate method to read content.

**Output:**

Emp Id	Name	Salary
1.0	John	2000000.0
2.0	Dean	420000.0
3.0	Sam	280000.0
4.0	Cass	6000000.0

### 3. Create New Excel File

Let us create a new excel file and write data in it. Following is the API which we will use for this purpose.

```
import org.apache.poi.hssf.usermodel.HSSFSheet;  
import org.apache.poi.hssf.usermodel.HSSFWorkbook;  
//..  
HSSFWorkbook workbook = new HSSFWorkbook();  
HSSFSheet sheet = workbook.createSheet("Sample sheet");  
//Create a new row in current sheet  
Row row = sheet.createRow(0);  
//Create a new cell in current row  
Cell cell = row.createCell(0);  
//Set value to new value  
cell.setCellValue("Blahblah");
```

Below is the complete code that writes a new excel with dummy data:

```
HSSFWorkbook workbook = new HSSFWorkbook();  
HSSFSheet sheet = workbook.createSheet("Sample sheet");  
  
Map<String, Object[]> data = new HashMap<String, Object[]>();  
data.put("1", new Object[] {"Emp No.", "Name", "Salary"});  
data.put("2", new Object[] {1d, "John", 1500000d});  
data.put("3", new Object[] {2d, "Sam", 800000d});  
data.put("4", new Object[] {3d, "Dean", 700000d});  
  
Set<String> keyset = data.keySet();  
int rownum = 0;  
for (String key : keyset) {  
    Row row = sheet.createRow(rownum++);  
    Object [] objArr = data.get(key);  
    int cellnum = 0;  
    for (Object obj : objArr) {  
        Cell cell = row.createCell(cellnum++);  
        if(obj instanceof Date)  
            cell.setCellValue((Date)obj);
```

```

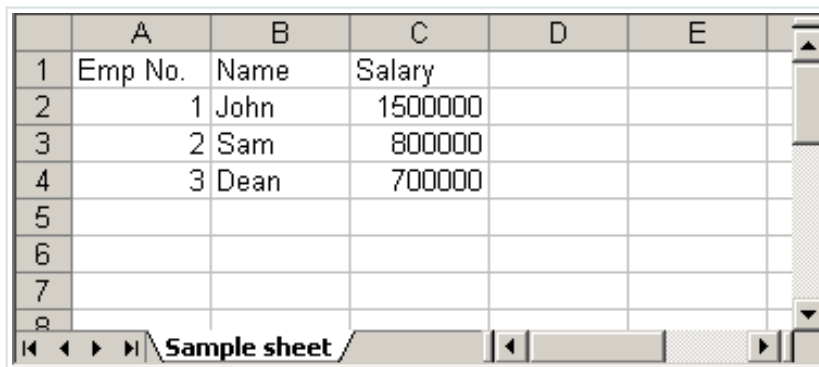
        else if(obj instanceof Boolean)
            cell.setCellValue((Boolean)obj);
        else if(obj instanceof String)
            cell.setCellValue((String)obj);
        else if(obj instanceof Double)
            cell.setCellValue((Double)obj);
    }
}

try {
    FileOutputStream out =
        new FileOutputStream(new File("C:\\new.xls"));
    workbook.write(out);
    out.close();
    System.out.println("Excel written successfully..");

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
}

```

**Output:** *new.xls*



The screenshot shows an Excel spreadsheet with a single sheet named 'Sample sheet'. The data is organized in a table with 5 columns: A (Emp No.), B (Name), C (Salary), D, and E. The first row (row 1) contains the headers: 'Emp No.', 'Name', 'Salary', and empty cells for D and E. The next three rows (rows 2, 3, and 4) contain employee data: Row 2 has '1', 'John', and '1500000'; Row 3 has '2', 'Sam', and '800000'; Row 4 has '3', 'Dean', and '700000'. Rows 5 through 8 are empty. The spreadsheet has a standard Excel interface with a grid, row numbers, column letters, and a status bar at the bottom.

	A	B	C	D	E
1	Emp No.	Name	Salary		
2	1	John	1500000		
3	2	Sam	800000		
4	3	Dean	700000		
5					
6					
7					
8					

## 4. Update Existing Excel File

Updating an existing excel file is straight forward. Open the excel using different API that we discussed



above and set the cell's value. One thing we need to note here is that we can update the excel file only when we close it first.

*update.xls*

	A	B	C	D	E
1	<b>Emp No.</b>	<b>Name</b>	<b>Salary</b>		
2	1	Bill	100		
3	2	Bobby	200		
4	3	Dick	300		
5					
6					

Following Java code read the above excel file and doubles the salary of each employee:

```
try {
    FileInputStream file = new FileInputStream(new File("C:\\update.xls"));

    HSSFWorkbook workbook = new HSSFWorkbook(file);
    HSSFSheet sheet = workbook.getSheetAt(0);
    Cell cell = null;

    //Update the value of cell
    cell = sheet.getRow(1).getCell(2);
    cell.setCellValue(cell.getNumericCellValue() * 2);
    cell = sheet.getRow(2).getCell(2);
    cell.setCellValue(cell.getNumericCellValue() * 2);
    cell = sheet.getRow(3).getCell(2);
    cell.setCellValue(cell.getNumericCellValue() * 2);

    file.close();

    FileOutputStream outFile = new FileOutputStream(new File("C:\\update.xls"));
    workbook.write(outFile);
    outFile.close();

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
```



```
e.printStackTrace();  
}
```

Steps to update excel file will be:

1. Open excel file in input mode (inputstream)
2. Use POI API and read the excel content
3. Update cell's value using different `setCellValue` methods.
4. Close the excel input file (inputstream)
5. Open same excel file in output mode (outputstream)
6. Write content of updated workbook in output file
7. Close output excel file

**Output:** *update.xls*

	A	B	C	D	E
1	Emp No.	Name	Salary		
2	1	Bill	200		
3	2	Bobby	400		
4	3	Dick	600		
5					
6					

## 5. Adding Formulas

Apache POI provides API to add excel formulas to cell programmatically. Following method that comes handy for this:

```
cell.setCellFormula("someformula")
```

For example:

```
cell.setCellFormula("A2*B2*C5")  
//or  
cell.setCellFormula("SUM(A1:A7)")
```

**Note:** Formula string should not start with equal sign (=)

Thus, following is incorrect way of adding formula:

```
cell.setCellFormula("=A2*B2*C5") //Ops! Won't work
```

The above code will throw:

```
org.apache.poi.ss.formula.FormulaParseException:  
The specified formula '=A2*B2*C5' starts with an equals sign which is not allowed
```

Following Java code creates a new excel sheet which calculates Simple Interest. It defines Principal amount, Rate of Interest and Tenure. We add an excel formula to calculate interest.

```
HSSFWorkbook workbook = new HSSFWorkbook();  
HSSFSheet sheet = workbook.createSheet("Calculate Simple Interest");  
  
Row header = sheet.createRow(0);  
header.createCell(0).setCellValue("Principal Amount (P)");  
header.createCell(1).setCellValue("Rate of Interest (r)");  
header.createCell(2).setCellValue("Tenure (t)");  
header.createCell(3).setCellValue("Interest (P r t)");  
  
Row dataRow = sheet.createRow(1);  
dataRow.createCell(0).setCellValue(14500d);  
dataRow.createCell(1).setCellValue(9.25);  
dataRow.createCell(2).setCellValue(3d);  
dataRow.createCell(3).setCellFormula("A2*B2*C2");  
  
try {  
    FileOutputStream out =  
        new FileOutputStream(new File("C:\\\\formula.xls"));  
    workbook.write(out);  
    out.close();  
    System.out.println("Excel written successfully..");  
}  
catch (FileNotFoundException e) {  
    e.printStackTrace();  
}  
catch (IOException e) {  
    e.printStackTrace();  
}  
}
```

Output: *formula.xls*

D2		fx =A2*B2*C2			
	A	B	C	D	E
1	Pricipal Amount (P)	Rate of Interest (r)	Tenure (t)	Interest (P r t)	
2	14500	9.25	3	402375	
3					
4					
5					
6					
7					
8					

Calculate Simple Interest

### Triggering Existing Excel Formulas

In certain cases your excel file might have formula defined and you may want to trigger those formulas since you updated it using POI. Following code snippet will do the trick.

```
FileInputStream fis = new FileInputStream("/somepath/test.xls");
Workbook wb = new HSSFWorkbook(fis); //or new XSSFWorkbook("C:\\test.xls")
FormulaEvaluator evaluator = wb.getCreationHelper().createFormulaEvaluator();
for(int sheetNum = 0; sheetNum < wb.getNumberOfSheets(); sheetNum++) {
    Sheet sheet = wb.getSheetAt(sheetNum);
    for(Row r : sheet) {
        for(Cell c : r) {
            if(c.getCellType() == Cell.CELL_TYPE_FORMULA) {
                evaluator.evaluateFormulaCell(c);
            }
        }
    }
}
```

We use `FormulaEvaluator` class to evaluate formula defined in each of the cell.

## 6. Adding Styles to Cell

Adding style to a cell is also piece of cake. Check following example which creates two new cell one with bold font and another with italic and add text to it.

```
HSSFWorkbook workbook = new HSSFWorkbook();
HSSFSheet sheet = workbook.createSheet("Style example");

HSSFFont font = workbook.createFont();
font.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);
HSSFCellStyle style = workbook.createCellStyle();
style.setFont(font);

Row row = sheet.createRow(0);
Cell cell = row.createCell(0);
cell.setCellValue("This is bold");
cell.setCellStyle(style);

font = workbook.createFont();
font.setItalic(true);
style = workbook.createCellStyle();
style.setFont(font);

row = sheet.createRow(1);
cell = row.createCell(0);
cell.setCellValue("This is italic");
cell.setCellStyle(style);

try {
    FileOutputStream out = new FileOutputStream(new File("C:\\style.xls"));
    workbook.write(out);
    out.close();
    System.out.println("Excel written successfully..");
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

Output: *style.xls*

	A	B	C	D
1	<b>This is bold</b>			
2	<i>This is italic</i>			
3				
4				
5				
6				
7				
8				

Style example

I hope this article is useful.

## Related Posts

- 1. [Read / Write CSV file in Java](#)
- 2. [Calculate Free Disk Space in Java using Apache Commons IO](#)
- 3. [Java: How to Load CSV file into Database](#)
- 4. [PDF Generation in Java using iText JAR](#)
- 5. [Creating ZIP and JAR Files in Java](#)
- 6. [Create JAR file in Java & Eclipse](#)
- 7. [How to Add Password Protection to PDF using iText in Java](#)

✉ **Get our Articles via Email. Enter your email address.**

Send Me Tutorials

Tags: [APACHE](#), [APACHE SOFTWARE FOUNDATION](#), [EXCEL](#), [JAVA](#)

## 118 Comments

« Previous 1 2 3



**Revathy**

5 August, 2013, 20:41

Hello 😊 I wanted to know how to compare 2 excel sheets??? pls help me ..ASAP !!!

[Reply](#)



**imrsn**

13 August, 2013, 19:30

Nice article, just one thing. Use LinkedHashMap instead of HashMap. Otherwise, when creating an excel file, users will get random rows. LinkedHashMap will take care of that.

[Reply](#)



**vinh**

14 August, 2013, 20:21

very good! thanks u!!

[Reply](#)



**Onu**

19 August, 2013, 19:47

Hi Patel

Thanks for a wonderful article, quick question – how I can ignore first row & print everything (since first row is the column title

[Reply](#)



**Yaakov**

20 August, 2013, 1:53

Really awesome! Exactly what I needed and very well explained. Just the basic information that I need to use the API 😊

[Reply](#)

[« Previous](#) [1](#) [2](#) [3](#)

## Leave a Reply

Your email address will not be published. Required fields are marked \*

**Name \***

Email \*

Website

Comment

#### Note

To post source code in comment, use `[code language] [/code]` tag, for example:

- `[code java] Java source code here [/code]`
- `[code html] HTML here [/code]`

**Post Comment**

#### Categories

[Home page](#) [Struts](#) [Spring](#) [AJAX](#) [PHP](#) [Java](#) [JavaEE](#)  
[JavaScript](#) [CSS](#) [Database](#) [Web 2.0](#) [News](#) [Fun](#)  
[General](#) [Featured](#) [Play Framework](#) [Android](#)  
[FreeMarker Template](#)

#### Site Pages

[About viralpatel.net](#) [Join Us](#) [Search](#) [Advertise](#)  
[Posts Feed](#) [Comments Feed](#)

[↑ Back to top](#)

Copyright © 2013 ViralPatel.net. All rights reserved :)