**Apache POI**

a project run by the [Apache Software Foundation](https://en.wikipedia.org/wiki/Apache_Software_Foundation), and previously a sub-project of the [Jakarta Project](https://en.wikipedia.org/wiki/Jakarta_Project), provides pure [Java](https://en.wikipedia.org/wiki/Java_platform) libraries for reading and writing files in [Microsoft Office](https://en.wikipedia.org/wiki/Microsoft_Office) [formats](https://en.wikipedia.org/wiki/File_format), such as [Word](https://en.wikipedia.org/wiki/Microsoft_Word), [PowerPoint](https://en.wikipedia.org/wiki/Microsoft_PowerPoint) and [Excel](https://en.wikipedia.org/wiki/Microsoft_Excel).

<http://www.tutorialspoint.com/apache_poi/apache_poi_workbooks.htm>

Create Blank Workbook

The following simple program is used to create a blank Microsoft Excel Workbook.

import java.io.\*;

import org.apache.poi.xssf.usermodel.\*;

public class CreateWorkBook

{

public static void main(String[] args)throws Exception

{

//Create Blank workbook

XSSFWorkbook workbook = new XSSFWorkbook();

//Create file system using specific name

FileOutputStream out = new FileOutputStream(

new File("createworkbook.xlsx"));

//write operation workbook using file out object

workbook.write(out);

out.close();

System.out.println("

createworkbook.xlsx written successfully");

}

}

Let us save the above Java code as CreateWorkBook.java, and then compile and execute it from the command prompt as follows:

$javac CreateWorkBook.java

$java CreateWorkBook

If your system environment is configured with the POI library, it will compile and execute to generate the blank Excel file named **createworkbook.xlsx** in your current directory and display the following output in the command prompt.

createworkbook.xlsx written successfully

Open Existing Workbook

Use the following code to open an existing workbook.

import java.io.\*;

import org.apache.poi.xssf.usermodel.\*;

public class OpenWorkBook

{

public static void main(String args[])throws Exception

{

File file = new File("openworkbook.xlsx");

FileInputStream fIP = new FileInputStream(file);

//Get the workbook instance for XLSX file

XSSFWorkbook workbook = new XSSFWorkbook(fIP);

if(file.isFile() && file.exists())

{

System.out.println(

"openworkbook.xlsx file open successfully.");

}

else

{

System.out.println(

"Error to open openworkbook.xlsx file.");

}

}

}

Save the above Java code as OpenWorkBook.java, and then compile and execute it from the command prompt as follows:

$javac OpenWorkBook.java

$java OpenWorkBook

It will compile and execute to generate the following output.

openworkbook.xlsx file open successfully.

After opening a workbook, you can perform read and write operations on it.

Read from a Spreadsheet

Let us consider the above excel file named **Writesheet.xslx** as input. Observe the following code; it is used for reading the data from a spreadsheet.

import java.io.File;

import java.io.FileInputStream;

import java.util.Iterator;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.xssf.usermodel.XSSFRow;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Readsheet

{

static XSSFRow row;

public static void main(String[] args) throws Exception

{

FileInputStream fis = new FileInputStream(

new File("WriteSheet.xlsx"));

XSSFWorkbook workbook = new XSSFWorkbook(fis);

XSSFSheet spreadsheet = workbook.getSheetAt(0);

Iterator < Row > rowIterator = spreadsheet.iterator();

while (rowIterator.hasNext())

{

row = (XSSFRow) rowIterator.next();

Iterator < Cell > cellIterator = row.cellIterator();

while ( cellIterator.hasNext())

{

Cell cell = cellIterator.next();

switch (cell.getCellType())

{

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();

}

fis.close();

}

}

Let us keep the above code in **Readsheet.java** file, and then compile and run it from the command prompt as follows:

$javac Readsheet.java

$java Readsheet

If your system environment is configured with the POI library, it will compile and execute to generate the following output in the command prompt.

EMP ID EMP NAME DESIGNATION

tp01 Gopal Technical Manager

tp02 Manisha Proof Reader

tp03 Masthan Technical Writer

tp04 Satish Technical Writer

tp05 Krishna Technical Writer

Types of Cells

The cell type specifies whether a cell can contain strings, numeric value, or formulas. A string cell cannot hold numeric values and a numeric cell cannot hold strings. Given below are the types of cells, their values, and type syntax.

|  |  |
| --- | --- |
| **Type of cell value** | **Type Syntax** |
| Blank cell value | XSSFCell.CELL\_TYPE\_BLANK |
| Boolean cell value | XSSFCell.CELL.TYPE\_BOOLEAN |
| Error cell value | XSSFCell.CELL\_TYPE\_ERROR |
| Numeric cell value | XSSFCell.CELL\_TYPE\_NUMERIC |
| String cell value | XSSFCell.CELL\_TYPE\_STRING |

This chapter takes you through the process of applying different formulas on cells using Java programming. The basic purpose of Excel application is to maintain numerical data by applying formulas on it.

In a formula, we pass dynamic values or locations of the values in the Excel sheet. On executing this formula, you get the desired result. The following table lists a few basic formulas that are frequently used in Excel.

|  |  |
| --- | --- |
| **Operation** | **Syntax** |
| Adding multiple numbers | = SUM(Loc1:Locn) or = SUM(n1,n2,) |
| Count | = COUNT(Loc1:Locn) or = COUNT(n1,n2,) |
| Power of two numbers | = POWER(Loc1,Loc2) or = POWER(number, power) |
| Max of multiple numbers | = MAX(Loc1:Locn) or = MAX(n1,n2,) |
| Product | = PRODUCT(Loc1:Locn) or = PRODUCT(n1,n2,) |
| Factorial | = FACT(Locn) or = FACT(number) |
| Absolute number | = ABS(Locn) or = ABS(number) |
| Today date | =TODAY() |
| Converts lowercase | = LOWER(Locn) or = LOWER(text) |
| Square root | = SQRT(locn) or = SQRT(number) |

The following code is used to add formulas to a cell and execute it.

import java.io.File;

import java.io.FileOutputStream;

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;

public class Formula

{

public static void main(String[] args)throws Exception

{

XSSFWorkbook workbook = new XSSFWorkbook();

XSSFSheet spreadsheet = workbook.createSheet("formula");

XSSFRow row = spreadsheet.createRow(1);

XSSFCell cell = row.createCell(1);

cell.setCellValue("A =" );

cell = row.createCell(2);

cell.setCellValue(2);

row = spreadsheet.createRow(2);

cell = row.createCell(1);

cell.setCellValue("B =");

cell = row.createCell(2);

cell.setCellValue(4);

row = spreadsheet.createRow(3);

cell = row.createCell(1);

cell.setCellValue("Total =");

cell = row.createCell(2);

// Create SUM formula

cell.setCellType(XSSFCell.CELL\_TYPE\_FORMULA);

cell.setCellFormula("SUM(C2:C3)" );

cell = row.createCell(3);

cell.setCellValue("SUM(C2:C3)");

row = spreadsheet.createRow(4);

cell = row.createCell(1);

cell.setCellValue("POWER =");

cell=row.createCell(2);

// Create POWER formula

cell.setCellType(XSSFCell.CELL\_TYPE\_FORMULA);

cell.setCellFormula("POWER(C2,C3)");

cell = row.createCell(3);

cell.setCellValue("POWER(C2,C3)");

row = spreadsheet.createRow(5);

cell = row.createCell(1);

cell.setCellValue("MAX =");

cell = row.createCell(2);

// Create MAX formula

cell.setCellType(XSSFCell.CELL\_TYPE\_FORMULA);

cell.setCellFormula("MAX(C2,C3)");

cell = row.createCell(3);

cell.setCellValue("MAX(C2,C3)");

row = spreadsheet.createRow(6);

cell = row.createCell(1);

cell.setCellValue("FACT =");

cell = row.createCell(2);

// Create FACT formula

cell.setCellType(XSSFCell.CELL\_TYPE\_FORMULA);

cell.setCellFormula("FACT(C3)");

cell = row.createCell(3);

cell.setCellValue("FACT(C3)");

row = spreadsheet.createRow(7);

cell = row.createCell(1);

cell.setCellValue("SQRT =");

cell = row.createCell(2);

// Create SQRT formula

cell.setCellType(XSSFCell.CELL\_TYPE\_FORMULA);

cell.setCellFormula("SQRT(C5)");

cell = row.createCell(3);

cell.setCellValue("SQRT(C5)");

workbook.getCreationHelper()

.createFormulaEvaluator()

.evaluateAll();

FileOutputStream out = new FileOutputStream(

new File("formula.xlsx"));

workbook.write(out);

out.close();

System.out.println("fromula.xlsx written successfully");

}

}

Save the above code as **Formula.java** and then compile and execute it from the command prompt as follows.

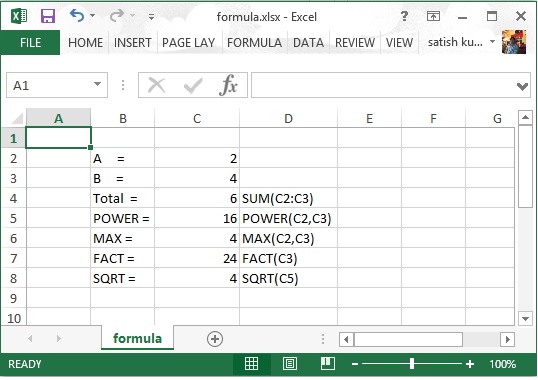
$javac Formula.java

$java Formula

It will generate an Excel file named **formula.xlsx** in your current directory and display the following output on the command prompt.

fromula.xlsx written successfully

The **formula.xlsx** file looks as follows.



[Best way to read a text file in Java?](http://stackoverflow.com/questions/4716503/best-way-to-read-a-text-file-in-java)

BufferedReader br = new BufferedReader(new FileReader("file.txt"));

try {

StringBuilder sb = new StringBuilder();

String line = br.readLine();

while (line != null) {

sb.append(line);

sb.append(System.lineSeparator());

line = br.readLine();

}

String everything = sb.toString();

} finally {

br.close();

}

try(BufferedReader br = new BufferedReader(new FileReader("file.txt"))) {

StringBuilder sb = new StringBuilder();

String line = br.readLine();

while (line != null) {

sb.append(line);

sb.append(System.lineSeparator());

line = br.readLine();

}

String everything = sb.toString();

}

**Using Apache Commons IO**

FileInputStream inputStream = new FileInputStream("foo.txt");

try {

String everything = IOUtils.toString(inputStream);

} finally {

inputStream.close();

}

And even simpler with Java 7:

try(FileInputStream inputStream = new FileInputStream("foo.txt")) {

Session IOUtils;

String everything = IOUtils.toString(inputStream);

}

The easiest way is to use the Scanner class in Java and the FileReader object. Simple example:

Scanner in = new Scanner(new FileReader("filename.txt"));

Here's another way to do it without using external libraries:

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

public String readFile(String filename)

{

String content = null;

File file = new File(filename); //for ex foo.txt

FileReader reader = null;

try {

reader = new FileReader(file);

char[] chars = new char[(int) file.length()];

reader.read(chars);

content = new String(chars);

reader.close();

} catch (IOException e) {

e.printStackTrace();

} finally {

if(reader !=null){reader.close();}

}

return content;

}

The methods within [org.apache.commons.io.FileUtils](http://commons.apache.org/proper/commons-io/apidocs/org/apache/commons/io/FileUtils.html) may also be very handy, e.g.:

/\*\*

\* Reads the contents of a file line by line to a List

\* of Strings using the default encoding for the VM.

\*/

static List readLines(File file)

* org.apache.poi.ss.usermodel
* interface Cell
  + **Method Summary**

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| [**CellRangeAddress**](https://poi.apache.org/apidocs/org/apache/poi/ss/util/CellRangeAddress.html) | [**getArrayFormulaRange**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getArrayFormulaRange())()  Only valid for array formula cells |
| boolean | [**getBooleanCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getBooleanCellValue())()  Get the value of the cell as a boolean. |
| int | [**getCachedFormulaResultType**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getCachedFormulaResultType())()  Only valid for formula cells |
| [**Comment**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Comment.html) | [**getCellComment**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getCellComment())()  Returns comment associated with this cell |
| java.lang.String | [**getCellFormula**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getCellFormula())()  Return a formula for the cell, for example, SUM(C4:E4) |
| [**CellStyle**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/CellStyle.html) | [**getCellStyle**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getCellStyle())()  Return the cell's style. |
| int | [**getCellType**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getCellType())()  Return the cell type. |
| int | [**getColumnIndex**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getColumnIndex())()  Returns column index of this cell |
| java.util.Date | [**getDateCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getDateCellValue())()  Get the value of the cell as a date. |
| byte | [**getErrorCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getErrorCellValue())()  Get the value of the cell as an error code. |
| [**Hyperlink**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Hyperlink.html) | [**getHyperlink**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getHyperlink())() |
| double | [**getNumericCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getNumericCellValue())()  Get the value of the cell as a number. |
| [**RichTextString**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/RichTextString.html) | [**getRichStringCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getRichStringCellValue())()  Get the value of the cell as a XSSFRichTextString |
| [**Row**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Row.html) | [**getRow**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getRow())()  Returns the Row this cell belongs to |
| int | [**getRowIndex**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getRowIndex())()  Returns row index of a row in the sheet that contains this cell |
| [**Sheet**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Sheet.html) | [**getSheet**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getSheet())()  Returns the sheet this cell belongs to |
| java.lang.String | [**getStringCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#getStringCellValue())()  Get the value of the cell as a string |
| boolean | [**isPartOfArrayFormulaGroup**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#isPartOfArrayFormulaGroup())() |
| void | [**removeCellComment**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#removeCellComment())()  Removes the comment for this cell, if there is one. |
| void | [**removeHyperlink**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#removeHyperlink())()  Removes the hyperlink for this cell, if there is one. |
| void | [**setAsActiveCell**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setAsActiveCell())()  Sets this cell as the active cell for the worksheet |
| void | [**setCellComment**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellComment(org.apache.poi.ss.usermodel.Comment))([**Comment**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Comment.html) comment)  Assign a comment to this cell |
| void | [**setCellErrorValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellErrorValue(byte))(byte value)  Set a error value for the cell |
| void | [**setCellFormula**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellFormula(java.lang.String))(java.lang.String formula)  Sets formula for this cell. |
| void | [**setCellStyle**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellStyle(org.apache.poi.ss.usermodel.CellStyle))([**CellStyle**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/CellStyle.html) style)  Set the style for the cell. |
| void | [**setCellType**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellType(int))(int cellType)  Set the cells type (numeric, formula or string). |
| void | [**setCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellValue(boolean))(boolean value)  Set a boolean value for the cell |
| void | [**setCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellValue(java.util.Calendar))(java.util.Calendar value)  Set a date value for the cell. |
| void | [**setCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellValue(java.util.Date))(java.util.Date value)  Converts the supplied date to its equivalent Excel numeric value and sets that into the cell. |
| void | [**setCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellValue(double))(double value)  Set a numeric value for the cell |
| void | [**setCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellValue(org.apache.poi.ss.usermodel.RichTextString))([**RichTextString**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/RichTextString.html) value)  Set a rich string value for the cell. |
| void | [**setCellValue**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setCellValue(java.lang.String))(java.lang.String value)  Set a string value for the cell. |
| void | [**setHyperlink**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Cell.html#setHyperlink(org.apache.poi.ss.usermodel.Hyperlink))([**Hyperlink**](https://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Hyperlink.html) link)  Assign a hyperlink to this cell |