# Apache POI to read write excel in Selenium Webdriver

## Table of content



## Apache POI with Selenium Webdriver Integration

In every Development environment, data is a mandatory part of their development; we put all your data in your program files. But if any change in data results in the editing of the program file, which makes us recompile the code and retest the compiled code.

If data Changes every day, are we going to edit the program file every day?
What happens if the compilation fails ?

Development should happen in such that data don't have any effect on the program files. Placing data outside the program is the only way to do it, like placing the data on excel files, property files, config files, JSON Files, XML files.

### Apache POI

Apache POI helps Java/Java related technologies to read and write Excel files on different platforms. Using apache poi, we can do read and write operation of both xls and xlsx file formats. Apache poi is an open-source tool developed by apache.

Apache POI will be helpful to modify the large content of data. Below is the step by step process to download Apache poi jar files.

Follow below steps to download Apache Poi:

*Open **https://poi.apache.org/***

*Click Downloads section on right side*

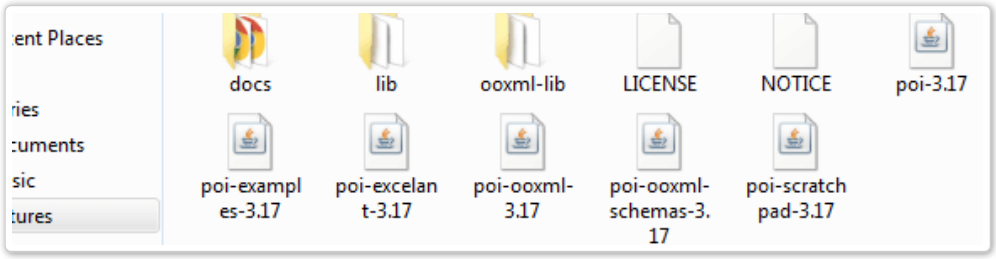

*Click on the poi-bin-######.zip link under binary distribution section*


*Click on the mirror link; apache would be suggesting the nearest location for your mirror link*

*Your poi files will be started to download*

*Extract the downloaded zip file*

*The extracted folder should contain the below files.*




**Headless browser in Selenium webdriver**

## Apache POI and Selenium Webdriver Integration

We can integrate apache poi with selenium webdriver to read and write excel files, and to make a data driver framework.

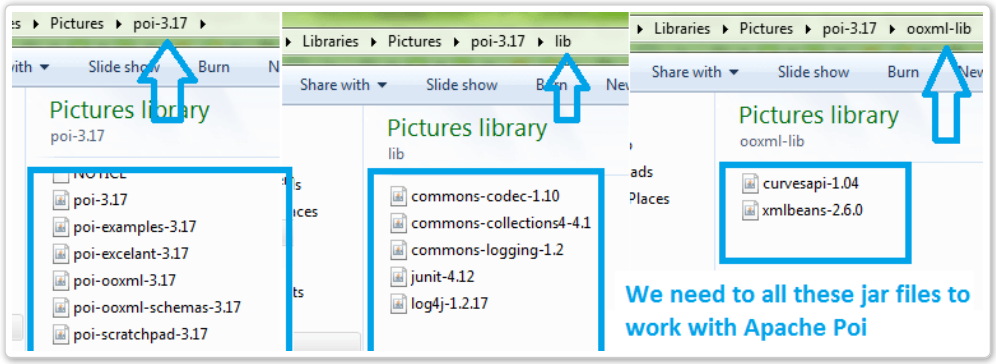
Follow below steps to integrate Apache Poi with selenium webdriver:

*Open eclipse selenium webdriver project*

*Right-click on the project and select Properties option*

*Click on Add External jars button*

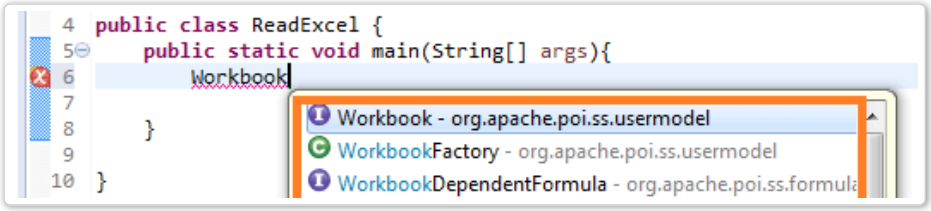
*Navigate to the folder where you have extracted apache poi zip file and add all the jars under main folder, jars under lib, jars under ooxml-lib*




*Click 'OK' on Build path Dialog after adding jar files*

*Create a Java Class under a package(files package here)*

*To test we have added apache poi successfully, type Workbook inside the main method and press CTRL+Space Bar, if you can see options from apache poi then integration is successful:*



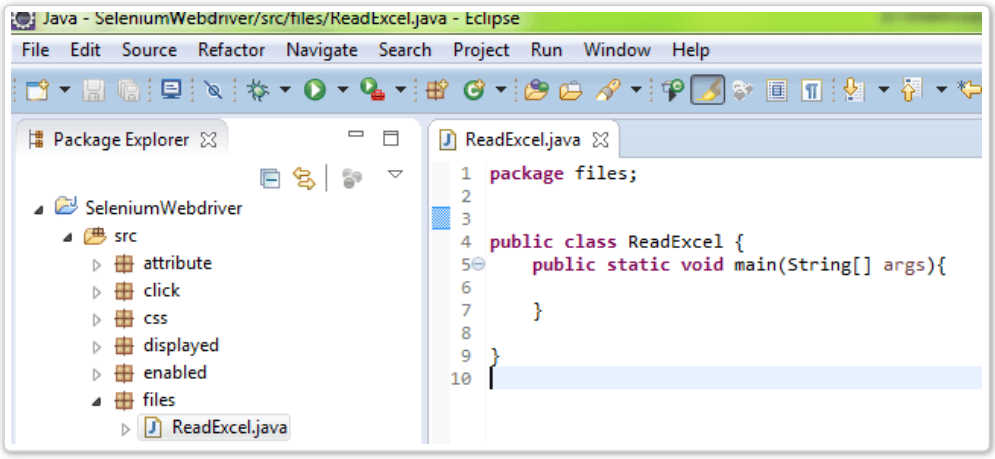

## Apache POI famous words

Apache POI excel library revolves around four key interfaces that actually represent the items in the excel file.
*Workbook: A workbook represents the excel file*
*Sheet: A workbook may contain many sheets. We can access the sheets either with a name or with an index.*
*Row: As the name suggests, It represents a row in the sheet.*
*Cell: A cell represents a column in the sheet.*

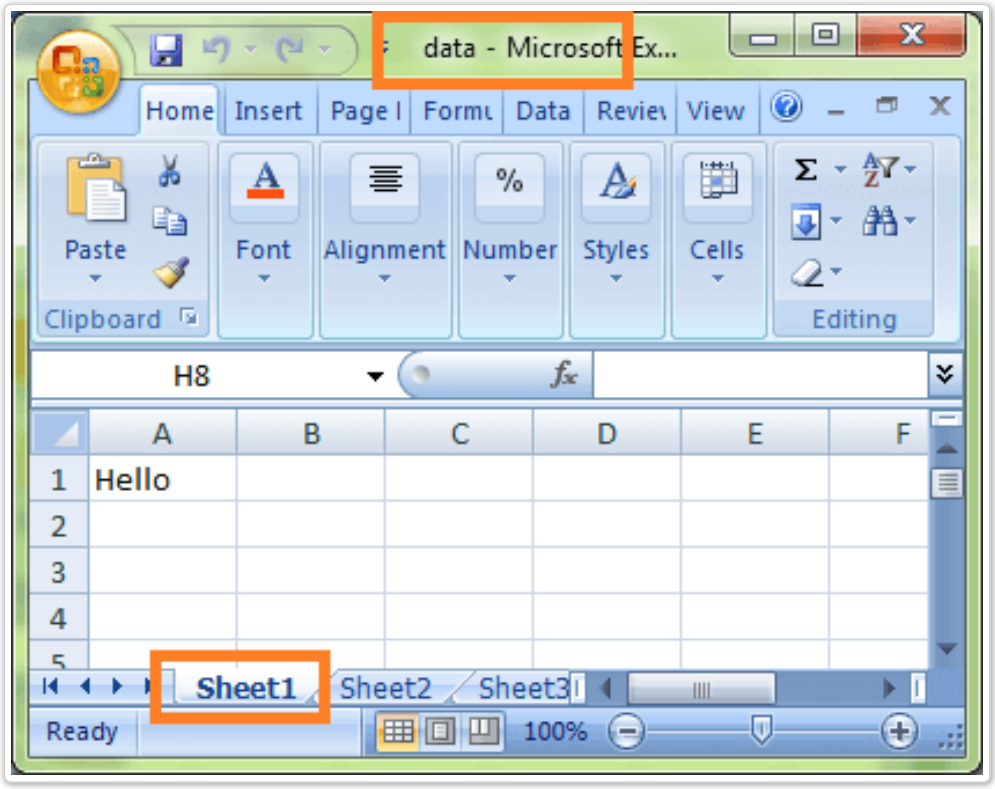
**Read & Write Barcode using Selenium Webdriver**.

# Read Excel with Apache POI

In this tutorial, we are going to see how to read data from the excel sheet and printing the value on the screen step by step. 1. Create a Java class on Eclipse




2. Create an excel sheet and place "Hello" in the top-right corner cell and save the file as data.xlsx




3. Create Object for FileInputStream with excel file path; it makes the file into a stream of input item

```
// create file input stream object for the excel sheet
FileInputStream fis = new FileInputStream("C:\pathX\data.xlsx");
```

4. Create a Workbook object by using create() method present in WorkbookFactory class, and pass the file Input stream object as a parameter.

```
// create object for work book
Workbook wb = WorkbookFactory.create(fis);
```

5. Create a Sheet object from the workbook object(wb); we have to pass the sheet name(Sheet1) as a parameter

```
//create object for sheet present in excel using Workbook object 'wb'
Sheet sheet = wb.getSheet("Sheet1");
```

6. Create a Row object from Sheet object(sheet); we have to pass Row number as a parameter

```
//create object for row present in sheet using Sheet object 'sheet'
```

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

7. Create a Cell object from the Row object(row); we have to pass Cell number as a parameter

```
//create object for cell present in row using Row object 'row'
Cell cell = row.getCell(0);
```

8. There is a method called getStringCellValue() in Cell Class, with the help of this method we can retrieve string cell value from the Cell,

```
//print the value present in the excel sheet
System.out.println(cell.getStringCellValue());
```

Complete Program to retrieve Cell value in Excel

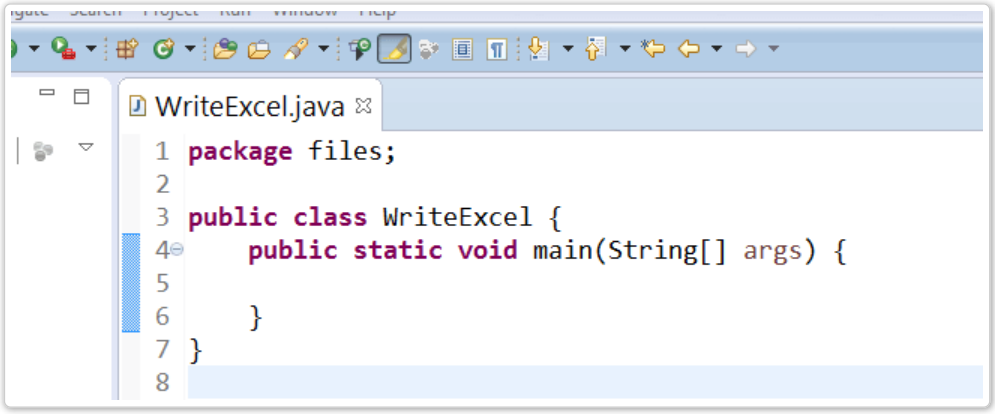
```
import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.EncryptedDocumentException;
import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class ReadExcel {
   public static void main(String[] args) throws Exception{
      // create file input stream object for the excel sheet
      FileInputStream fis = new FileInputStream("C:\pathXX\data.xlsx");
      // create object for work book
      Workbook wb = WorkbookFactory.create(fis);
      //create object for sheet present in excel using Workbook object 'wb'
      Sheet sheet = wb.getSheet("Sheet1");
      //create object for row present in sheet using Sheet object 'sheet'
      Row row = sheet.getRow(0);
      //create object for cell present in row using Row object 'row'
      Cell cell = row.getCell(0);
      //print the value present in the excel sheet
      System.out.println(cell.getStringCellValue());
   }
}
```

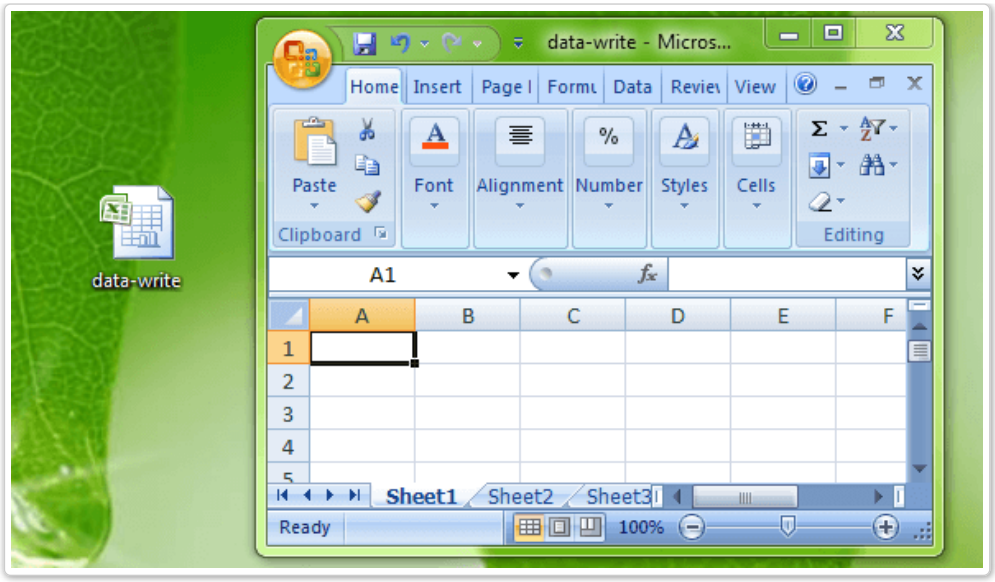
```
Output : Hello
```

## Write Excel with Apache POI

In this tutorial, we are going to see how to write data to excel sheet step by step detail. 1. Create a Java class on Eclipse




2. Create an excel file on your local machine ( you can find system created file bottom section ).




3. Create Object for FileInputStream with excel file path, and it makes the file into a stream of input item

```
// create file input stream object for the excel sheet
FileInputStream fis = new FileInputStream("C:\pathX\data-write.xlsx");
```

4. Create a Workbook object by using create() method present in WorkbookFactory class, and pass the file Input stream object as a parameter.

```
// create object for work book
Workbook wb = WorkbookFactory.create(fis);
```

5. Create a Sheet object from a workbook object(wb); we have to pass the sheet name(Sheet1) as a parameter

```
//create object for sheet present in excel using Workbook object 'wb'
Sheet sheet = wb.getSheet("Sheet1");
```

6. Create a Row object from Sheet object(sheet); we have to pass Row number as a parameter

```
//create object for row present in sheet using Sheet object 'sheet'
Row row = sheet.getRow(0);
```

7. Create a Cell object from Row object(row); here we have to user createCell() method to create a cell in particular row, we have to pass Cell number as a parameter

```
//create object for cell present in row using Row object 'row'
Cell cell = row.createCell(0);
```

8. There is a method called setCellValue() in Cell Class, with the help of this method we can write cell value to the Cell in excel file,

```
//print the value present in the excel sheet
cell.setCellValue("Written by apache poi");
```

9. create an object for FileOutputStream which makes the content of excel into output stream items, so that we can write an excel,

```
//creates file output stream
FileOutputStream fos = new FileOutputStream("C:\pathX\data-write.xlsx");
```

10. With the help of write() present in the Workbook class we can write the excel into the local system, this method accepts FileOutputStream object as a parameter

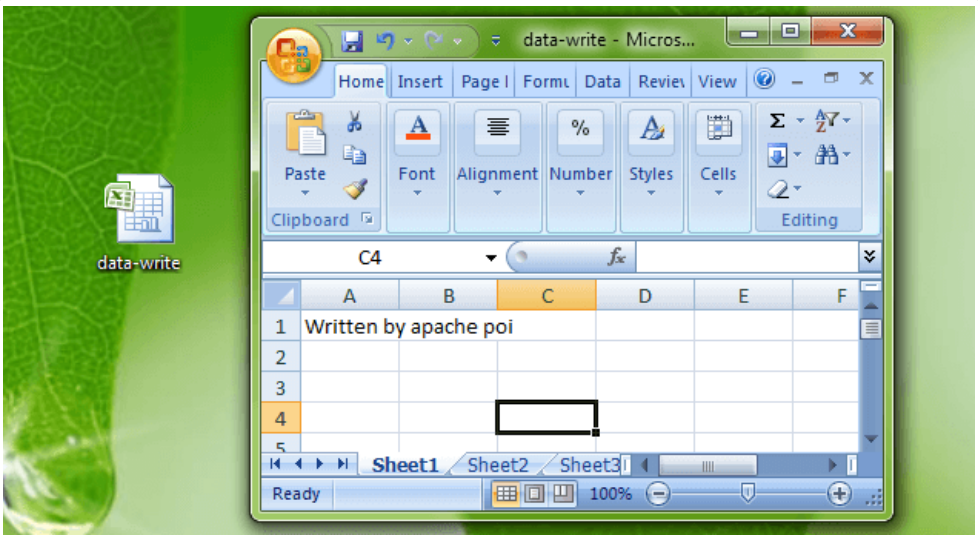
```
//writes excecl sheet into local file system
```

```
wb.write(fos);
```

Complete Program to Write Cell value in Excel.

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
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.ss.usermodel.WorkbookFactory;

public class WriteExcel {
   public static void main(String[] args) throws Exception {
      // create file input stream object for the excel sheet
      FileInputStream fis = new FileInputStream("C:\path\data-write.xlsx");
      // create object for work book
      Workbook wb = WorkbookFactory.create(fis);
      //create object for sheet present in excel using Workbook object 'wb'
      Sheet sheet = wb.getSheet("Sheet1");
      //create object for row present in sheet using Sheet object 'sheet'
      Row row = sheet.getRow(0);
      //create object for cell present in row using Row object 'row'
      Cell cell = row.createCell(0);
      cell.setCellValue("Written by apache poi");
      FileOutputStream fos = new FileOutputStream("C:\path\data-write.xlsx");
      wb.write(fos);
   }
}
```




## Retrieving Cell values by CellType

We can retrieve cell value using the getStringCellValue method, but it only works for String values. In actual; day to day activities, we may store more types of data in excel sheets like Number, boolean, strings.

We have different methods to retrieve different types of data. in apache poi

To retrieve different data, you may check each cell's type and then extract its value using various type-specific methods.

You should understand that the below methods will not extract a type of data from the cell when you store a particular data type in a cell, then the total cell is of

that type.

So these methods will fetch total value present in the cell.

*getBooleanCellValue() - To fetch boolean data from the excel*

*getDateCellValue() - fetches date values from the cell*

*getNumericCellValue() - fetches numeric value*

*getCellFormula() - fetches the data from the formula cell.*

```
public static void printCellValue(Cell cell) {
   switch (cell.getCellTypeEnum()) {
      case BOOLEAN:
         System.out.print(cell.getBooleanCellValue());
         break;
      case STRING:
         System.out.print(cell.getRichStringCellValue().getString());
         break;
      case NUMERIC:
         if (DateUtil.isCellDateFormatted(cell)) {
            System.out.print(cell.getDateCellValue());
         } else {
            System.out.print(cell.getNumericCellValue());
         }
         break;
      case FORMULA:
         System.out.print(cell.getCellFormula());
         break;
      case BLANK:
         System.out.print("");
         break;
      default:
         System.out.print("");
   }
}
```