ASPOSE
Your File Format Experts

CASE STUDIES

Aspose.Cells for Java Case Study

Advizeo by setec / Setec Smart Efficiency

Using Aspose Cells for Java to generate automatic internal reports for IOT sensors and energy savings & consumptions

Pierre-Eric OUDIN, Chief Operating & Technology Officer, 24/01/2018

About Setec Smart Efficiency

Setec smart efficiency is a young French Start-up based in Paris.

We are incorporated in the SETEC group, a big French engineering office at the origin of some major French infrastructures like the Channel tunnel, the engineering of many highspeed railroads, bridges etc…

At setec smart efficiency, our goals are to :

- ensure our clients are making energy savings ;
- satisfy the client with the comfort in his installations (monitoring humidity, temperature, CO2…);
- Help maintenance teams to detect bad behaviours in the installed material;

All these goals are reachable by putting some connected non-intrusive measure instruments providing real-time data on behalf of available IOT networks and exploiting it by using our software "Advizeo by setec". Our teams are composed of energy managers and developers.

Problem

Our energy managers are accompanying our clients who are using our "Saas solution" called Advizeo. But in order to provide a higher level of energy reporting and make clients realizing energy savings, they would like to have **specific internal tools** in order to save time which is today allocated to making some Excel-based reports.

Solution

The solution we found to solve our energy managers problem was to conceive a software which will produce automatically the data reporting based on the data we produce through our API.

The prerequesite for the software was that **we had to use the existing Excel templates** our energy managers were filling and formatting by hand :

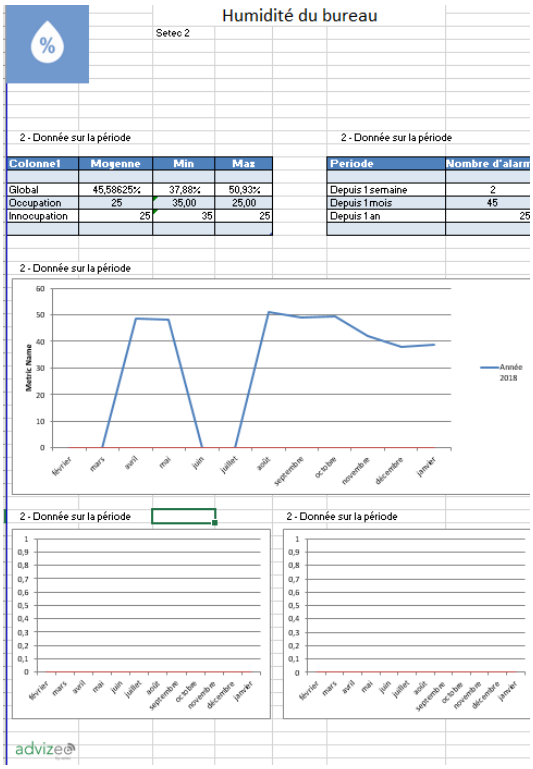


Figure 1 - Example of excel file used to summarize the data for a sensor

So in order to respect this input, **we went looking for a technical solution**, a library that could meet **our requirements**:

- Take Excel file as input ;
- Allow inserting and manipulating pictures ;
- Allow executing formulas stored in the Excel file ;
- Allow data insertion ;
- Allow chart configuration and display ;
- Allow playing with cell formatting ;

- Allow exporting as PDF ;
- Library available in Java ;
- Library which does not require MS Office suite installed because app will be deployed on a Linux server.

With the help of Aspose.Cells for Java, we were able to integrate it in a Java console Application (using Java 1.8).

The Application connects to the database and retrieve data about buildings, then an Excel template is opened and we use Aspose.Cells Java API to interact with the opened Workbook.

```
Workbook newWorkbookBuilding = null;
Cells cellsDataSensors;

try {
    newWorkbookBuilding = new Workbook(filename);
} catch (Exception e) {
    e.printStackTrace();
}

//Clear the range B2:E10002 of the worksheet "Data_sensors"
Worksheet newWorksheetDataSensors = newWorkbookBuilding.getWorksheets().get("Data_sensors");
cellsDataSensors = newWorksheetDataSensors.getCells();
cellsDataSensors.clearContents( startRow: 1,  startColumn: 1,  endRow: 10000,  endColumn: 4);

//Set the name of the new workbook.
String nameNewWorkbookBuilding = "Energy_report_" + building.name.replace(" ", "_");

//Write the name of the building in the cell H2 in the worksheet "Data_sensors".

cellNameBuildingDataSensors = cellsDataSensors.get( row: 1,  column: 7);
cellNameBuildingDataSensors.setValue(building.name);
```

Figure 2 - Clearing some data in the opened template and setting building's name information.

After making many other operations we are able to re-compute all formulas and save our workbook as a new Excel file and also as a PDF file. We can also disable some sheets we do not want to display : it is very useful for the PDF output !

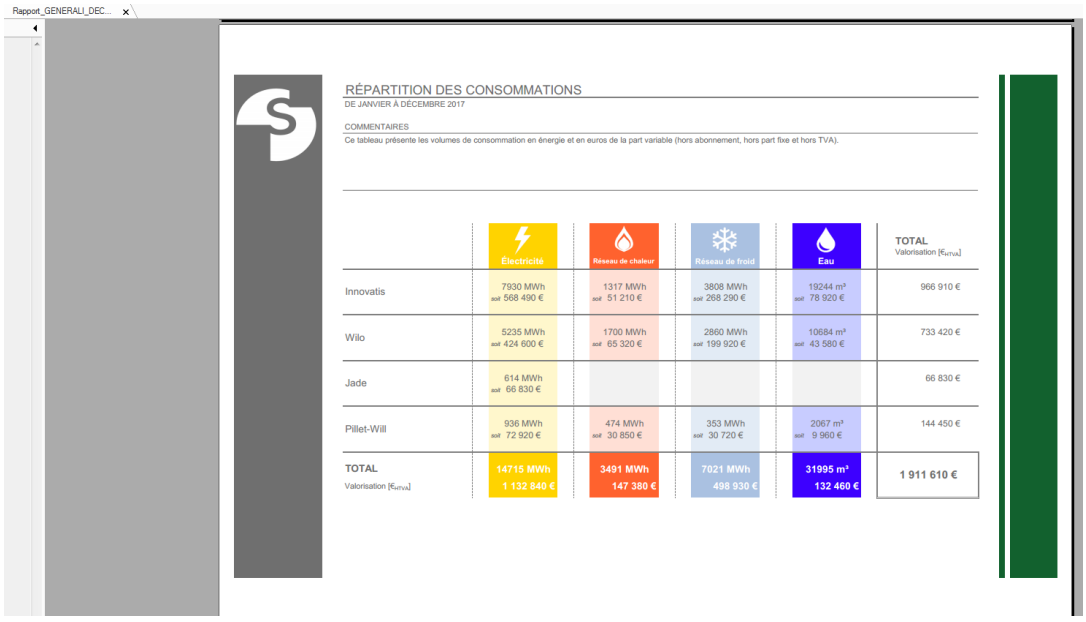
```java
    //This method permits to save the energy report.
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    private void save(Workbook newWorkbookBuilding,
                      String folderEnergyReport,
                      String nameNewWorkbookBuilding
    ) {
        try {
            if (newWorkbookBuilding != null) {
                //Mask the worksheets "PrepareHistogramme", "PrepareLine", "Pres", "Data_sensors", "PivotTable_sensors".
                WorksheetCollection newWorksheetsCollection = newWorkbookBuilding.getWorksheets();
                newWorksheetsCollection.get("PrepareHistogram").setVisible(false);
                newWorksheetsCollection.get("PrepareLine").setVisible(false);
                newWorksheetsCollection.get("PresLine").setVisible(false);
                newWorksheetsCollection.get("PresHistogram").setVisible(false);
                newWorksheetsCollection.get("Data_sensors").setVisible(false);
                //newWorksheetsCollection.get("PivotTable_sensors").setVisible(false);

                //Save the new workbook for each building in Excel file.
                ////////////////////////////////////////////////////////////////////////////////////////////////////////
                //Save formulas for each workbook.
                newWorkbookBuilding.calculateFormula();
                newWorkbookBuilding.save( fileName: folderEnergyReport + nameNewWorkbookBuilding + ".xlsx");
                System.out.println("File " + nameNewWorkbookBuilding + " saved in Excel file!");
                ////////////////////////////////////////////////////////////////////////////////////////////////////////

                //Save the new workbook for each building in PDF file.
                ////////////////////////////////////////////////////////////////////////////////////////////////////////
                PdfSaveOptions pdfSaveOptions = new PdfSaveOptions();
                pdfSaveOptions.setCalculateFormula(true);
                newWorkbookBuilding.save( fileName: folderEnergyReport + nameNewWorkbookBuilding + ".pdf", pdfSaveOptions);
                System.out.println("File " + nameNewWorkbookBuilding + " saved in PDF file!");
                ////////////////////////////////////////////////////////////////////////////////////////////////////////
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Figure 3 - Example of file saving method using Aspose.Cells for Java

At the end of the process the files are available to the energy managers in our network filesystem.



RÉPARTITION DES CONSOMMATIONS

DE JANVIER À DÉCEMBRE 2017

COMMENTAIRES

Ce tableau présente les volumes de consommation en énergie et en euros de la part variable (hors abonnement, hors part fixe et hors TVA).

	Electricité	Réseau de chaleur	Réseau de froid	Eau	TOTAL Valorisation [€HTVA]
Innovatis	7930 MWh soit 568 490 €	1317 MWh soit 51 210 €	3808 MWh soit 268 290 €	19244 m³ soit 78 920 €	966 910 €
Wilo	5235 MWh soit 424 600 €	1700 MWh soit 65 320 €	2860 MWh soit 199 920 €	10684 m³ soit 43 580 €	733 420 €
Jade	614 MWh soit 66 830 €				66 830 €
Pillet-Will	936 MWh soit 72 920 €	474 MWh soit 30 850 €	353 MWh soit 30 720 €	2067 m³ soit 9 960 €	144 450 €
TOTAL Valorisation [€HTVA]	14715 MWh 1 132 840 €	3491 MWh 147 380 €	7021 MWh 498 930 €	31995 m³ 132 460 €	1 911 610 €

Figure 4 - Another file generated as pdf ouput

Experience

Finding a solution: We first gave a look at Aspose because one of our IT Project managers has used Aspose in a company he used to work before. But we also made a comparison between others well known tools like Crystal Reports or Jasper reports. But all these tools did not meet the target process we wanted to implement. They did not match the technical environment we wanted to use and furthermore they would have required the use of a new report editing tool that our business teams didn't know. That is why Aspose.Cells for Java was our better option. We first use the trial version and knew we would like to take a licence in order to take advantages of watermark removal and the support.

Implementation: We first launch a POC with a Freelance person in order to test if our solution was good, then an internal developer took charge of the industrialisation of the process. The POC took one week because we had a lot of internal data to retrieve in the report but the adaptation to Aspose.Cell API for Java took only a few hours in a workday.

As a developer, it is very convenient to have a library through which you can make operations in an Excel file as if you were coding VBA Macros : the cells indexes are the same. Moreover, interaction with the Excel files comes very easy like :

- Get a specific datasheet by its name (e.g : myWorksheetCollection.get("worksheetName") ;
- Setting a specific cell value (e.g : myCell.get(rowIndex, colIndex).setValue(myValue));
- Interacting through a worksheet's pictures collection (e.g : myWorksheet.getPictures()) ;
- Specifying charts datasource for vertical/horizontal axes : (e.g : chart.getNSeries().get(serieIndex).setXValues(worksheetDataRange));

Another thing which is great is that cell ranges can be called as in Excel formulas (e.g : "E8:F14" or "SheetName!E5"), so migrating from VBA to Java code if there was any macros inside the template file is also possible.

Outcome: We succeeded in testing the solution on different datasets but we are still limited by the trial version with only 100 opened files.

Next Steps

As our initial problem was solved, we are planning to put in place more different report types using excel inputs and using Aspose.Cells for Java in these new projects but with using the Java Software to bin the data to the report.

We also plan to buy a licence of Aspose.Cells.for Java in order to replace the trial version and benefit from all the features without limitations because trial version put watermarks on pdf and excel files and the limit for opened files is 100.

Summary

We would recommend the use of Aspose.cells for Java because it is easy to implement and the API is simple to understand and very fluent. The only thing we perhaps regret is the price of the library which is not very affordable for a start-up. Nevertheless, for a company with a higher budget it will not be a problem.