Apache POI i Commons CSV

Maksim Dziatkou, Gabriel Filipowicz

Apache POI

Apache POI

- Zbiór open-source bibliotek Java do obsługi plików Microsoft Office (Excel, PowerPoint i Word).
- Jest popularnym wyborem w aplikacjach, które potrzebują automatyzować przetwarzanie dokumentów Office, w celu generowania raportów i analizowania danych z Excela.

POI - Poor Obfuscation Implementation.
Odnosi się do sarkastycznego stwierdzenia, że Microsoft wprowadził zabezpieczenia utrudniające inżynierię odwrotną, jednak nie były one wystarczająco skomplikowane, aby uniemożliwić stworzenie narzędzia, które sobie z nimi poradzi.

Instalacja (obsługa plików Excel)

Maven

poprzez dodanie zależności w pom.xml

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.3.0</version>
</dependency>
```

Pure Java

dodanie plików .jar do External Libraries
https://poi.apache.org/download.html

Kluczowe moduły

- **HSSF** (Horrible Spreadsheet Format) do obsługi plików w formacie .xls
- **XSSF** (XML Spreadsheet Format) do obsługi plików w formacie .xlsx (nowsze wersje Excela, czyli od 2007)

Ograniczenia

HSSF - wykresy nie są obsługiwane
XSSF - zużywa dużo pamięci
https://poi.apache.org/components/spreadsheet/limitations.html

HSSF: tworzenie arkusza kalkulacyjnego

```
//create workbook
Workbook workbook = new HSSFWorkbook(); //Excel '97(-2007) file format
//create sheet
Sheet sheet = workbook.createSheet( s: "Sheet1");
//create a row in the sheet
Row row0 = sheet.createRow( i: 0);
//add sells to the sheet
row0.createCell( i: 0);
//set value to the cell
row0.getCell( i: 0).setCellValue("your value =)");
```

HSSF: odczyt arkusza kalkulacyjnego z pliku

```
try {
    // Reading file from local directory
    FileInputStream file = new FileInputStream(new File( pathname: "excel_read.xls"));
    // Create Workbook instance holding reference to .xls file
    var workbook = new HSSFWorkbook(file);

    var sheet = workbook.getSheetAt( index: 0);
    var row = sheet.getRow( rowIndex: 1);
    var cell = row.getCell( cellnum: 1);
    System.out.println(cell.getStringCellValue());
    cell = row.getCell( cellnum: 2);
    System.out.println(cell.getNumericCellValue());

    file.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

HSSF: tworzenie pliku .xls

```
//generate .xls file
try {
    FileOutputStream out = new FileOutputStream(
            new File( pathname: "excel_file.xls"));
    workbook.write(out);
    out.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

HSSF: Przykład 1

	A	B	C	D	E
1	No.	Name	Age	E-mail	Balance
2	1	John William	53	william.john@gmail.com	75000.00
3	2	Piter Parker	25	parker.piter@gmail.com	2000
4					
5					

```
row0.createCell( i: 0).setCellValue("No.");
row0.createCell( i: 1).setCellValue("Name");
row0.createCell( i: 2).setCellValue("Age");
row0.createCell( i: 3).setCellValue("E-mail");
row0.createCell( i: 4).setCellValue("Balance");

//create the 1st row
Row row1 = sheet.createRow( i: 1);
//insert data in the first row
row1.createCell( i: 0).setCellValue(1);
row1.createCell( i: 1).setCellValue("John William");
row1.createCell( i: 2).setCellValue(53);
row1.createCell( i: 3).setCellValue("william.john@gmail.com");
row1.createCell( i: 4).setCellValue("75000.00");

//create the 2nd row
Row row2 = sheet.createRow( i: 2);
//insert data in the second row
row2.createCell( i: 0).setCellValue(2);
row2.createCell( i: 1).setCellValue("Piter Parker");
row2.createCell( i: 2).setCellValue(25);
row2.createCell( i: 3).setCellValue("parker.piter@gmail.com");
row2.createCell( i: 4).setCellValue(2000.00);
```

HSSF: Styles

	A	B	C	D	E
1	No.	Name	Age	E-mail	Balance
2	1	John William	53	william.john@gmail.com	75000.00
3	2	Piter Parker	25	parker.piter@gmail.com	2000
4					
5					

```
//styles
CellStyle cellStyle = workbook.createCellStyle();
cellStyle.setAlignment(HorizontalAlignment.CENTER);
cellStyle.setVerticalAlignment(VerticalAlignment.CENTER);
cellStyle.setWrapText(true);
cellStyle.setFillForegroundColor(IndexedColors.LIGHT_ORANGE.getIndex());
cellStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND);
cellStyle.setBorderTop(BorderStyle.THIN);
cellStyle.setBorderLeft(BorderStyle.NONE);
cellStyle.setBorderRight(BorderStyle.MEDIUM);
cellStyle.setBorderBottom(BorderStyle.MEDIUM_DASH_DOT);

Font font = workbook.createFont();
font.setFontName("Courier New");
font.setFontHeightInPoints((short) 14);
font.setBold(true);
font.setItalic(true);
cellStyle.setFont(font);

row0.getCell( i: 0).setCellStyle(cellStyle);
row0.getCell( i: 1).setCellStyle(cellStyle);
row0.getCell( i: 2).setCellStyle(cellStyle);
row0.getCell( i: 3).setCellStyle(cellStyle);
row0.getCell( i: 4).setCellStyle(cellStyle);
```

HSSF: Przykład 2

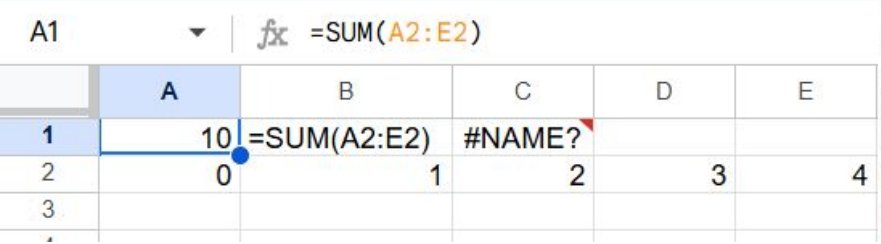
```
var persons = new ArrayList<Person>();
persons.add(new Person( name: "Jim Carrey", age: 62,
        balance: 888888.88f, email: "jcarrey@student.agh.edu.pl"));
persons.add(new Person( name: "Tom", age: 3, balance: 10f,
        email: "cat@gmail.com"));
persons.add(new Person( name: "Jerry", age: 2, balance: 58.31f,
        email: "mouse@gmail.com"));
persons.add(new Person( name: "Jaroslaw Jarzabkowski", age: 36,
        balance: 999999f, email: "pashabiceps@gmail.com"));

int index = 3;
for(Person person : persons) {
    Row row = sheet.createRow(index);
    row.createCell( i: 0).setCellValue(index);
    row.createCell( i: 1).setCellValue(person.getName());
    row.createCell( i: 2).setCellValue(person.getAge());
    row.createCell( i: 3).setCellValue(person.getEmail());
    row.createCell( i: 4).setCellValue(person.getBalance());
    index++;
}
```

HSSF: Przykład 2

	A	B	C	D	E
1	No.	Name	Age	E-mail	Balance
2	1	John William	53	william.john@gmail.com	75000.00
3	2	Piter Parker	25	parker.piter@gmail.com	2000
4	3	Jim Carrey	62	jcarrey@student.agh.edu.pl	888888,875
5	4	Tom	3	cat@gmail.com	10
6	5	Jerry	2	mouse@gmail.com	58,31000137
7	6	Jaroslaw Jarzabkowski	36	pashabiceps@gmail.com	999999
8					
9					

HSSF: Formuły




```java
Workbook workbook = new HSSFWorkbook();
Sheet sheet = workbook.createSheet( s: "Sheet1");
Row row = sheet.createRow( i: 0);
Cell cell = row.createCell( i: 0);
// You should NOT add a "=" to the front of the string
// Only commas may be used to separate arguments
cell.setCellFormula("SUM(A2:E2)");
Cell cell1 = row.createCell( i: 1);
cell1.setCellValue("=SUM(A2:E2)"); //string value, not formula
Cell cell2 = row.createCell( i: 2);
//POI only supports English function-names
cell2.setCellFormula("CYMM(A2:E2)"); //won't work

row = sheet.createRow( i: 1);
for(int i = 0; i < 5; i++){
    cell = row.createCell(i);
    cell.setCellValue(i);
}
```

HSSF: Formula evaluator

Wykonywanie i obliczanie formuł

Gdy komórka z formułą generuje błąd, FormulaEvaluator zwraca obiekt CellValue z typem ERROR (cellValue.getCellType() == ERROR)

```
row = sheet.getRow( i: 0);
cell = row.getCell( i: 0);

//System.out.println(cell.getNumericCellValue()); //exception
//System.out.println(cell.getStringCellValue()); // exception

System.out.println(cell.getCellType());
//output: FORMULA

FormulaEvaluator evaluator = workbook.getCreationHelper().createFormulaEvaluator();
CellValue cellValue = evaluator.evaluate(cell);
System.out.println(cellValue);
//output: org.apache.poi.ss.usermodel.CellValue [10.0]
System.out.println(cellValue.getNumberValue());
//output: 10.0
```

XSSF: Charts (1)

```
try(XSSFWorkbook workbook = new XSSFWorkbook()){
    XSSFSheet sheet = workbook.createSheet( sheetname: "Sheet1");
    //ys
    Row row = sheet.createRow( rownum: 1);
    row.createCell( i: 0).setCellValue("y:");
    row.createCell( i: 1).setCellValue(8);
    row.createCell( i: 2).setCellValue(13);
    row.createCell( i: 3).setCellValue(4);
    row.createCell( i: 4).setCellValue(22);
    row.createCell( i: 5).setCellValue(15);
    //xs
    row = sheet.createRow( rownum: 2);
    row.createCell( i: 0).setCellValue("x:");
    row.createCell( i: 1).setCellValue(1);
    row.createCell( i: 2).setCellValue(2);
    row.createCell( i: 3).setCellValue(3);
    row.createCell( i: 4).setCellValue(4);
    row.createCell( i: 5).setCellValue(5);
```

XSSF: Charts (2)

```
//To start drawing you need to call createPatriarch
//This has the effect erasing any other shape information stored in that sheet
XSSFDrawing drawing = (XSSFDrawing) sheet.createDrawingPatriarch();
XSSFClientAnchor anchor = drawing.createAnchor( dx1: 0,  dy1: 0,  dx2: 0,  dy2: 0,  col1: 0,  row1: 5,  col2: 10,  row2: 20);

//Create chart
XSSFChart chart = drawing.createChart(anchor);
chart.setTitleText("my chart");
chart.setTitleOverlay(false);
XDDFChartLegend legend = chart.getOrAddLegend();
legend.setPosition(LegendPosition.TOP_RIGHT);

//Create axes
XDDFCategoryAxis bottomAxis = chart.createCategoryAxis(AxisPosition.BOTTOM);
bottomAxis.setTitle("x");
XDDFValueAxis leftAxis = chart.createValueAxis(AxisPosition.LEFT);
leftAxis.setTitle("f(x)");
leftAxis.setCrosses(AxisCrosses.AUTO_ZERO);
```

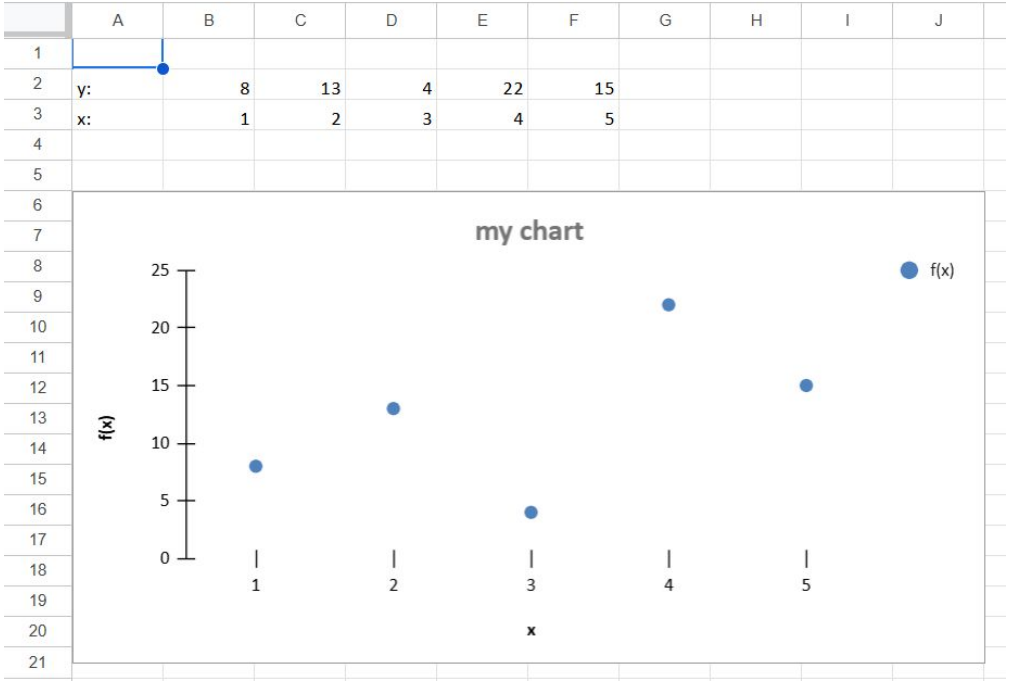
XSSF: Charts (3)

```
//Create data sources
XDDFNumericalDataSource<Double> xs = XDDFDataSourcesFactory.fromNumericCellRange(sheet,
        new CellRangeAddress( firstRow: 1,  lastRow: 1,  firstCol: 1,  lastCol: 5));
XDDFNumericalDataSource<Double> ys = XDDFDataSourcesFactory.fromNumericCellRange(sheet,
        new CellRangeAddress( firstRow: 2,  lastRow: 2,  firstCol: 1,  lastCol: 5));

//Set chart data
XDDFChartData data = chart.createData(ChartTypes.SCATTER, bottomAxis, leftAxis);
//Add data sources
XDDFChartData.Series series1 = data.addSeries(ys, xs);
series1.setTitle( title: "f(x)",  titleRef: null);
//Create plot
chart.plot(data);

//Save .xlsx file
try (FileOutputStream fileOut = new FileOutputStream( name: "chart.xlsx")) {
    workbook.write(fileOut);
}
```

XSSF: Charts


A B C D E F G H I J
1
2 y: 8 13 4 22 15
3 x: 1 2 3 4 5
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
my chart
f(x)
25
20
15
10
5
0
f(x)
1
2
3
4
5
x

Dokumentacja

Przykłady:

https://poi.apache.org/components/spreadsheet/examples.html

Przykład kodu z wykresem:
https://svn.apache.org/repos/asf/poi/trunk/poi-examples/src/main/java/org/apache/poi/examples/xssf/usermodel/BarChart.java

Żródła:

Busy Developers' Guide to Features
https://poi.apache.org/components/spreadsheet/quick-guide.html

Commons CSV

Commons CSV

Common CSV jest to biblioteka do Javy od Apache Commons. Powstała aby ułatwić pracę z plikami CSV. Dodajemy do projektu poprzez dodanie zależności w pliku pom.xml. Wszystkie wersje Apache Commons CSV oraz innych bibliotek można znaleźć na stronie mvn repository.

```
<dependencies>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-csv</artifactId>
        <version>1.12.0</version>
    </dependency>
```

Budowa Common CSV

Zawiera wiele wygodnych funkcjonalności takie jak manipulacja nagłówkami czy praca z różnymi ogranicznikami danych. Biblioteka zapewnia możliwość obsługi standardowych formatów CSV, ale obsługuje również dostosowanie do formatów o różnych strukturach.

Najważniejsze z klas Apache Commons CSV tp:

- **CSV Format** - definiuje liczne możliwości dostosowania działania pozostałych klas
- **CSV Parser** - do odczytu danych
- **CSV Printer** - do zapisu danych

CSVFormat

Klasa CSV Format odgrywa kluczową rolę w bibliotece CSV Apache Commons, ponieważ określa sposób formatowania lub analizowania danych CSV. CSVFormat udostępnia kilka predefiniowanych formatów, które są zgodne z powszechnymi standardami CSV.

Type	Field
CSVFormat	DEFAULT
CSVFormat	EXCEL
CSVFormat	INFORMIX_UNLOAD
CSVFormat	INFORMIX_UNLOAD_CSV
CSVFormat	MONGODB_CSV
CSVFormat	MONGODB_TSV
CSVFormat	MYSQL
CSVFormat	ORACLE
CSVFormat	POSTGRESQL_CSV
CSVFormat	POSTGRESQL_TEXT
CSVFormat	RFC4180
CSVFormat	TDF

Edycja CSV Format

Każdy format można edytować stosując metodę builder oraz odwołania do pojedynczych funkcjonalności. Najważniejsze z nich to:

- .withDelimiter(‘znak’)
- .withHeader(‘tablica nagłówków’)
- .withFirstRecordAsHeader()
- .withQuote('znak')

The `CSVFormat.Builder` settings are:

- `setDelimiter(',')`
- `setQuote('"')`
- `setRecordSeparator("\r\n")`
- `setIgnoreEmptyLines(true)`
- `setDuplicateHeaderMode(DuplicateHeaderMode.ALLOW_ALL)`

```
CSVFormat csvFormat = CSVFormat.DEFAULT.builder()
        .setHeader("author", "title")
        .setSkipHeaderRecord(true)
        .build();
```

Odczytywanie plików CSV

Jednym ze sposobów na odczytywanie plików CSV jest skorzystanie z klasy Reader w połączeniu z CSV Parser. Przy wczytywaniu plików najważniejszą rzeczą jest prawidłowe sformatowanie danych.

Najważniejsze funkcje:

- .parse(nazwa pliku, CSVFormat)
- .getRecords()
- .iterator()
- CSVRecord.get(nazwa nagłówka / indeks)

Dostosowanie wczytywanych typów

```
CSVFormat csvFormat = CSVFormat.DEFAULT.builder()
        .setDelimiter('/')
        .setHeader()
        .setSkipHeaderRecord(true)
        .build();

try (CSVParser parser = new CSVParser(new FileReader(file), csvFormat)) {
    for (CSVRecord record : parser) {
        String author = record.get(1);
        String title = record.get(2);
        tab.add(Integer.parseInt(record.get(0)));
        values.add(Double.parseDouble(record.get(3)));
        AUTHOR_BOOK_MAP.put(author, title);
    }
```

Stosowanie .setQuote

```
id,name,description
1,"John Doe","Software engineer, specializes in backend"
2,"Jane Smith","Data analyst, expert in SQL"
```

```
1 John Doe Software engineer, specializes in backend
2 Jane Smith Data analyst, expert in SQL

Process finished with exit code 0
```

```
CSVFormat csvFormat = CSVFormat.DEFAULT.builder()
        .setDelimiter(',')
        .setQuote('"')
        .setHeader("id", "name", "description")
        .setSkipHeaderRecord(true)
        .build();


Iterable<CSVRecord> records = csvFormat.parse(in);
for (CSVRecord record : records) {

    String author = record.get("name");
    Integer x = Integer.parseInt(record.get("id"));
    String title = record.get("description");
    System.out.println(x + " " + author + " " + title);
```

Obsługa niestandardowych plików CSV

W plikach CSV możemy się spotkać z bardziej skomplikowanymi przypadkami odczytu. Są to pliki z:

- z rekordami z niespójną ilością kolumn
- osadzonymi cudzysłowami
- polami wielowierszowymi

```
try (CSVParser parser = CSVFormat.DEFAULT
        .withAllowMissingColumnNames()
        .withQuote( quoteChar: '"')
        .withEscape('"')
        .withRecordSeparator('\n')
        .withIgnoreEmptyLines()
        .parse(new FileReader(file))) {

    for (CSVRecord record : parser) {
        System.out.println(record);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Zapis danych do pliku CSV

Klasa CSVPrinter zapewnia elastyczne wprowadzanie danych do pliku. Dane można zapisywać wiersz po wierszu, za pomocą list, tablic lub pojedynczych pól. Klasa wspiera dodawanie nagłówków do plików, co jest użyteczne przy ustrukturyzowanym imporcie danych do aplikacji.

Najważniejsze funkcje:

- .printRecord(lista wartości)
- .print(pojedyncza wartość)
- .flush() - opróżnia klasę i zapewnia poprawny zapisach następnych danych. Używane przy długotrwałych procesach.

```java
public class CSVPrinterExample {  no usages
    public void Write(String file) {  no usages
        String[] headers = {"ID", "Name", "Age"};
        String[][] data = {
                {"1", "Alice", "30"},
                {"2", "Bob", "24"},
                {"3", "Charlie", "29"}};

        try (FileWriter out = new FileWriter(file);
             CSVPrinter printer = new CSVPrinter(out, CSVFormat.DEFAULT.withHeader(headers))) {

            for (String[] row : data) {
                printer.printRecord(Arrays.asList(row));}

            System.out.println("CSV file created successfully.");

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

Obsługa wyjątków

Apache Commons CSV nie zapewnia wbudowanego odzyskiwania uszkodzonych plików, ale umożliwia wychwytywanie i rejestrowanie wyjątków podczas przetwarzania każdego wiersza(możliwe za pomocą składni try i catch).

Rodzaje najczęstszych błędów:

- IllegalArgumentException - występuje, gdy wczytywane wiersze są niezgodne z opisem w CSVFormat
- IOException - występuje, gdy strumień prowadzi do nie istniejącego pliku lub nie można się do niego dostać.

Przykład obsługi wyjątków

```
public void example () {  no usages
    try (FileReader reader = new FileReader( fileName: "Plik.csv");
         CSVParser parser = CSVFormat.DEFAULT.withHeader().parse(reader)) {

        for (CSVRecord record : parser) {
            try {
                String id = record.get(0);
                String name = record.get(1);
                String age = record.get(2);
                System.out.printf("ID: %s, Name: %s, Age: %s%n", id, name, age);
            } catch (IllegalArgumentException e) {
                System.err.println("Record format error: " + e.getMessage());
            }
        }

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

Źródła

- https://commons.apache.org/proper/commons-csv/user-guide.html
- https://www.baeldung.com/apache-commons-csv