Apache POI
Apache Commons CSV

Anna Moroń
Mateusz Mrowiec
Iwo Ropa

Apache POI

Apache POI

- Apache POI to biblioteka stworzona przez Apache Software Foundation podobnie jak Maven
- POI jako projekt ma tworzyć API do manipulacji plikami formatów opartych na OOXML i OLE2
- POI jako biblioteka pozwala na odczytywanie i tworzenie plików MS Excel, MS Word oraz MS Powerpoint. Najbardziej rozwiniętą funkcjonalnością tej biblioteki jest ta związana z MS Excel (formaty HSSF i XSSF). Prace dalej się ciągną nad formatami związanymi z MS Word i MS Powerpoint

Instalacja

Aby zacząć używanie Apache POI, wystarczy dodać dependency do Apache POI Commons i Apache POI API based on OPC and OOXML Schemas np. ze strony mvnrepository.com.

```
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>5.3.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.3.0</version>
</dependency>
```

Tworzenie nowego workbooka

```
Workbook wb = new HSSFWorkbook();
try  (OutputStream fileOut = new FileOutputStream("workbook.xls")) {
   wb.write(fileOut);
}
```

```
Workbook wb = new XSSFWorkbook();
try (OutputStream fileOut = new FileOutputStream("workbook.xlsx")) {
   wb.write(fileOut);
}
```

Tworzenie nowego arkusza

```
Workbook wb = new HSSFWorkbook(); // or new XSSFWorkbook();

Sheet sheet1 = wb.createSheet("new sheet");

Sheet sheet2 = wb.createSheet("second sheet");


try (OutputStream fileOut = new FileOutputStream("workbook.xls")) {

   wb.write(fileOut);

}
```

Wczytywanie danych z arkusza

```
Workbook wb = WorkbookFactory.create(new File("MyExcel.xls"))

DataFormatter formatter = new DataFormatter();

Sheet sheet1 = wb.getSheetAt(0);

for (Row row : sheet1) {

   for (Cell cell : row) {

        CellReference cellRef = new CellReference(row.getRowNum(), cell.getColumnIndex());

        System.out.print(cellRef.formatAsString());

        System.out.print(" - ");

        String text = formatter.formatCellValue(cell);

        System.out.println(text);

   }

}
```

Wpisywanie danych do arkusza

```
try (InputStream inp= new FileInputStream("workbook.xls")){
   Workbook wb1=WorkbookFactory.create(inp);
   Sheet sheet=wb1.getSheetAt(0);
   Row row = sheet.getRow(2);
   Cell cell = row.getCell(3);
   if(cell==null){
       cell = row.createCell(3);
       cell.setCellType(CellType.STRING);
       cell.setCellValue("a test");
       try (OutputStream fileOut =new FileOutputStream("workbook.xls")){
           wb1.write(fileOut);
       }
   }

}
```

Formatowanie Komórek

- Wyrównywanie
  - setAligment(halign) - ustawienie wyrównania tekstu horyzontalnego
  - setVerticalAligment(valign) - wyrównanie wyrównania wertykalnego
- Dodawanie krawędzi
  - CellStyle style=wb.createCellStyle();
  - style.setBorder@Bottom/Right/Left/Top(BorderStyle.THIN) - ustawienie odpowiedniej krawędzi
  - style.set@Bottom/Right/Left/Top@BorderColor(IndexedColors.BLACK.getIndex()) - ustawia kolor odpowiedniej krawędzi

Dodawanie formuł

```
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);
```

Dodawanie formuł cd.

```
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);
```

Apache Commons CSV

Apache Commons CSV

Biblioteka zapewniająca prosty interfejs do wczytywania i zapisywania plików CSV w różnych standardach/formatach.

Licencja: Apache 2.0

Repozytorium: https://github.com/apache/commons-csv

Instalacja

Najłatwiej zainstalować z mavena:

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

Aktualną wersję można sprawdzić na mavenrepository.com.

Można tez oczywiście pobrac gotowe buildy biblioteki z https://commons.apache.org/csv/download_csv.cgi.

Proste wczytywanie plików CSV

```
final CSVFormat csvFormat = CSVFormat.DEFAULT;

try (CSVParser csvParser = new CSVParser(new FileReader(filePath), csvFormat)) {

   for (CSVRecord csvRecord : csvParser) {
       for (String value : csvRecord) {
           System.out.print(value);
           System.out.print("\t");
       }
       System.out.println();
   }

} catch (IOException e) {
   System.err.println(e.getMessage());
}
```

Wczytanie z nagłówkiem

```
final CSVFormat csvFormat = CSVFormat.DEFAULT.builder().setHeader("h1", "h2", "h3").build();

try (CSVParser csvParser = new CSVParser(new FileReader(filePath), csvFormat)) {

    for (CSVRecord csvRecord : csvParser) {
        System.out.println(csvRecord.get("h1"));
        System.out.println(csvRecord.get("h3"));
    }

} catch (IOException e) {
    System.err.println(e.getMessage());
}
```

Wczytanie z nagłówkiem z Enuma

```
enum Header {
   H1, H2, H3
}

final CSVFormat csvFormat = CSVFormat.DEFAULT.builder().setHeader(Header.class).build();

try (CSVParser csvParser = new CSVParser(new FileReader(filePath), csvFormat)) {

   for (CSVRecord csvRecord : csvParser) {
       System.out.println(csvRecord.get(Header.H1));
       System.out.println(csvRecord.get(Header.H3));
   }

} catch (IOException e) {
   System.err.println(e.getMessage());
}
```

Zapewnione formaty

Biblioteka zapewnia pewne standardowe ustawienia formatów takie jak:

DEFAULT, EXCEL, INFORMIX_UNLOAD, INFORMIX_UNLOAD_CSV, MONGODB_CSV, MONGODB_TSV, MYSQL, ORACLE, POSTGRESQL_CSV, POSTGRESQL_TEXT, RFC4180, TDF

Są one zdefiniowane jako stałe na klasie CSVFromat więc można ich łatwo użyć:

```
CSVFormat format = CSVFormat.EXCEL;
```

Klasa CSVFormat - Builder

Obiekty klasy CSVFormat posiadają opcje modyfikacji jak i tworzenia nowych formatów przy pomocy buildera.

```
CSVFormat csvFormatWrite = CSVFormat.Builder.create() // utworzenie w formacie DEFAULT
                // można też np. CSVFormat.EXCEL.builder() lub CSVFormat.newFormat('\t').builder()
        .setTrim(true) // true - pominięcie pustych rekordów z przodu i tyłu
        .setHeader("Name", "Date", "Status")
        .setSkipHeaderRecord(false)
        .setEscape('*')
        .setQuote('@') // zamiast domyślnego znaku ", aby pominąć .setQuote(null)
        .setNullString("nic")
        .setDelimiter("|")
        .setRecordSeparator("\n") // wejście może zawierać separatory '\n', '\r' and "\r\n"
        .build();
```

Pełna lista: https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVFormat.Builder.html#setQuote(java.lang.Character)

Klasa CSVParser

parse() - metoda statyczna służąca do tworzenia tych obiektów z

getRecords() - zwraca listę rekordów

getHeaderNames() - zwraca listę nazw nagłówków

getRecordNumber() - zwraca liczbę rekordów (nie działa z wielolinijkowy polami)

Cała dokumentacja: https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVParser.html

Klasa CSVRecord

get(key) - zwraca wartość pola o podanym indeksie lub jeśli podano String lub Enum to odpowiadające mu pole zgodnie z nagłówkiem

isConsistent() - sprawdza czy ilość pól zgadza się z tym co jest zdefiniowane w nagłówku.

isSet(key) - sprawdza czy istnieje odpowiednie pole (można podać int jako index lub Stringa jako nazwę nagłówka)

Cała dokumentacja: https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVRecord.html

Zapis do pliku CSV

```
final CSVFormat csvFormat = CSVFormat.DEFAULT;

try (CSVPrinter csvPrinter = new CSVPrinter(new FileWriter(filePath), csvFormat)) {

   csvPrinter.printRecord("a", "b", "c");
   csvPrinter.printRecord("e", "f", "g");

   csvPrinter.flush();
} catch (IOException e) {
   System.err.println(e.getMessage());
}
```

Klasa CSVPrinter

printRecord(data) - zapisuje jeden rekord. Argumentem mogą być poszczególne wartości (konwertowalne do Stringa) lub Stream lub Iterator.

printRecords(data) - zapisuje wiele rekordów. Argumentem mogą być poszczególne wartości lub Stream lub Iterator, które w zależności od tego czy są podwójnie zagnieżdżone zostaną dodane jako rekordy.

Cała dokumentacja: https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVPrinter.html

Przykład printRecords

```
final String[][] data = {
   {"a", "b", "c"},
   {"d", "e", "f"},
   {"g", "h", "i"},
};

csvPrinter.printRecords(data);
/*
a,b,c
d,e,f
g,h,i
*/


final String[] data = {"a","b","c"};

csvPrinter.printRecords(data);
/*
a
b
c
*/
```

Ciekawostka

```
final String[][] data = {
   {"a", "b", "c"},
   {"a", "b", "c"},
   {"a", "b", "c"},
};

csvPrinter.printRecord(data);
/*
[Ljava.lang.String;@19469ea2,[Ljava.lang.String;@1c20c684,[Ljava.lang.String;@1fb3ebeb
*/
```

Bibliografia

- https://poi.apache.org/
- https://commons.apache.org/
- https://mvnrepository.com/artifact/org.apache.commons/commons-csv
- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml/5.3.0
- https://mvnrepository.com/artifact/org.apache.poi/poi
- https://poi.apache.org/components/spreadsheet/quick-guide.html