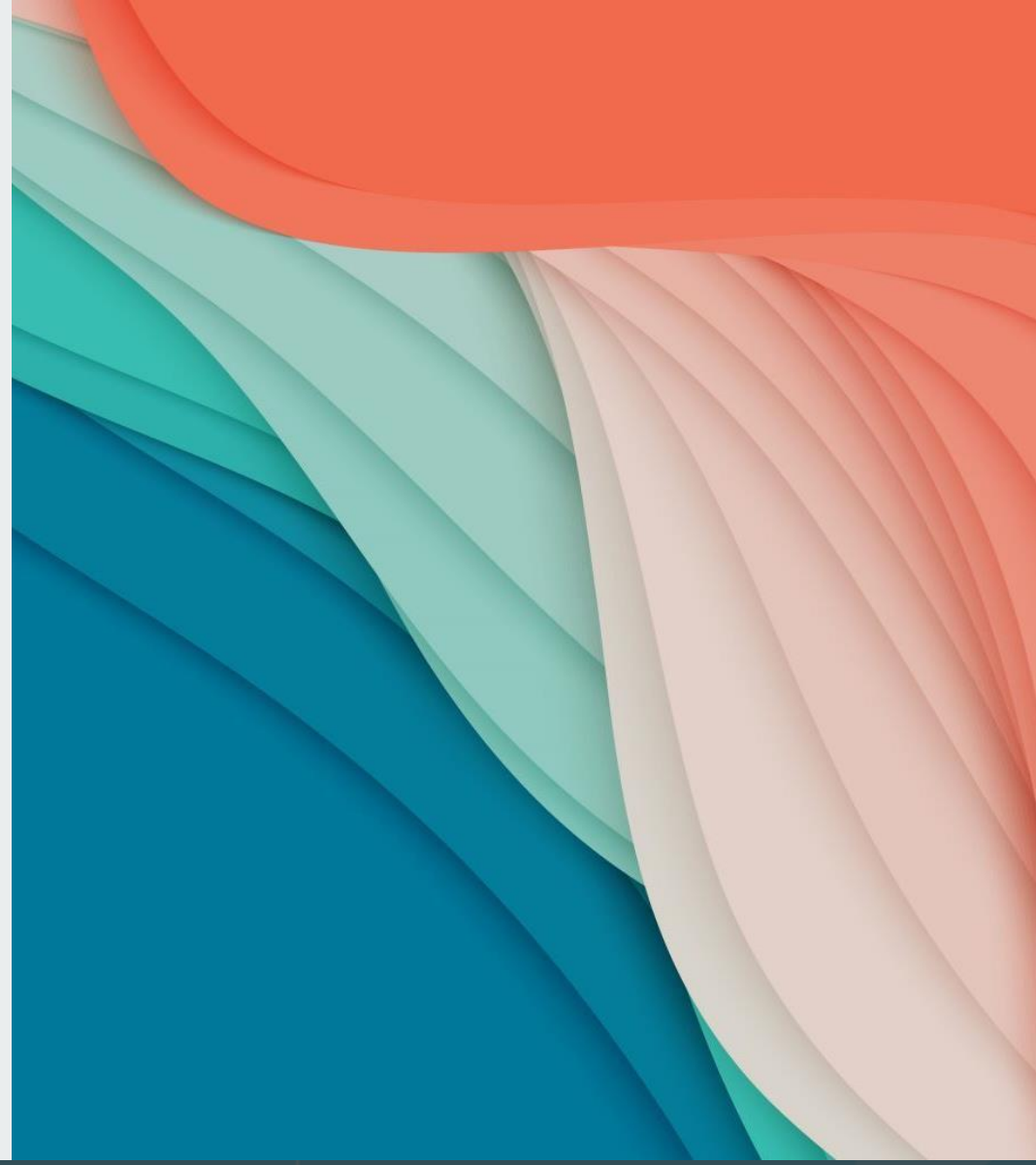
Apache POI i Apache Commons CSV

Kacper Gawroński, Łukasz Garczyński, Jan Jędra



Apache POI

Czym jest Apache POI?

- Apache POI jest biblioteką typu open source stworzoną przez Apache Software Foundation.
- Służy ona do odczytu, edycji oraz tworzenia plików pakietu Microsoft Office, typu Excel, Word, oraz PowerPoint z poziomu Javy. Posiada również obsługę dużo starszych formatów Microsoft Office takich jak xls.
- Każdy rodzaj pliku pakietu Microsoft Office ma dedykowany komponent.
- Najczęściej używa się jej ze względu na tworzeniu plików xslx. co umożliwia pisanie raportów oraz przeprowadzania testowania sterowanego danymi.

Architektura Apache POI

Apache POI podzielona jest na wiele części, z których każda z nich zajmuje się innym formatem pakietu Microsoft Office. Najważniejsze z nich to:

- XSSF – zajmuje się plikami typu xlsx. czyli plikami utworzonymi w Excelu.
- XWPF – zajmuje się plikami typu docx. czyli tymi, które tworzymy w Wordzie.
- XSLF – zajmuje się plikami typu pptx. które powstają w programie PowerPoint
- POIFS – nie używany tak jak inne, służy jako budulec pozostałych istniejących klas.

Jak pobrać Apache POI?

Proces instalacji Apache POI jest naprawdę trywialny:

1. Jeżeli używamy Maven jest to tylko kwestia dodania odpowiednich do naszych potrzeb dependency, gdyż każda z nich zawiera komponenty, które odpowiadają za część formatów plików pakietu Microsoft Office :

Component	Application type	Maven artifactId
POIFS	OLE2 Filesystem	*poi*
HPSF	OLE2 Property Sets	*poi*
HSSF	Excel XLS	*poi*
HSLF	PowerPoint PPT	*poi-scratchpad*
HWPF	Word DOC	*poi-scratchpad*
HDGF	Visio VSD	*poi-scratchpad*
HPBF	Publisher PUB	*poi-scratchpad*
HSMF	Outlook MSG	*poi-scratchpad*
DDF	Escher common drawings	*poi*
HWMF	WMF drawings	*poi-scratchpad*
OpenXML4J	OOXML	*poi-ooxml* plus either *poi-ooxml-lite* or *poi-ooxml-full*
XSSF	Excel XLSX	*poi-ooxml*
XSLF	PowerPoint PPTX	*poi-ooxml*
XWPF	Word DOCX	*poi-ooxml*
XDGF	Visio VSDX	*poi-ooxml*
Common SL	PowerPoint PPT and PPTX	*poi-scratchpad* and *poi-ooxml*
Common SS	Excel XLS and XLSX	*poi-ooxml*

Pobieranie Apache POI cd.

Nas interesuje jedynie dependency poi-ooxml, które umożliwia nam działanie na plikach z rozszerzeniami między innymi xlsx. docx. oraz pptx. :

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

Jeżeli ktoś jednak nie jest fanem Maven, Apache POI można pobrać w bardziej przyziemny sposób. Wchodząc na oficjalną stronę Apache możemy pobrać plik zip. który zawiera w sobie wszystkie potrzebne elementy, by zacząć korzystać z Apache POI. Po jego pobraniu wystarczy go rozpakować i wszystkie pliki przenieść do projektu, w którym chcemy użyć Apache POI.

Wady i zalety używania Apache POI do pracy z arkuszami kalkulacyjnymi

Zalety:

- Posiada wsparcie rozszerzeń xls. jak i xlsx.
- Umożliwia pracę przy rozbudowanych arkuszach kalkulacyjnych
- Pozwala na wykorzystanie bardziej zaawansowanych funkcji arkusza

Wady:

- Nie obsługuje makr, jednak zachowuje je podczas odczytywania arkusza oraz umożliwia ekstrakcję makr z niego
- Przy tworzeniu wykresów ogranicza się jedynie do prostych zmian, sam umożliwia tworzenie tylko niektórych wykresów
- Potrafi wykorzystywać dużą ilość pamięci

Podstawowy podział interfejsów w Apache POI - Excel

Jeżeli chodzi o tworzenie arkuszów kalkulacyjnych, to do dyspozycji mamy dwa moduły : HSSF oraz XSSF. Ten pierwszy umożliwia pracę jedynie przy formacie xls. który nie jest już używany, ale został zostawiony ze względu na kompatybilność. Natomiast XSSF służy do pracy z rozszerzeniem xlsx.. Podstawowymi budulcami, które wykorzystuje się przy tworzeniu arkusza kalkulacyjnego są interfejsy:

- Workbook – odpowiada skoroszytowi w Excelu
- Spreadsheet – jest to pojedynczy arkusz
- Row – wiersz w arkuszu
- Cell - komórka wchodząca w skład wiersza

Rodzaje interfejsów w Apache POI - Excel

Zanim zaczniemy tworzyć pierwszy arkusz kalkulacyjny przy pomocy Javy i Apache POI trzeba najpierw zastanowić się, której implementacji wcześniej wymienionych interfejsów użyjemy. Możemy użyć trzech rozwiązań:

Interface	Apache POI XLSX	Apache POI HSSF	Apache POI SXSSF
Workbook	XSSFWorkbook	HSSFWorkbook	SXSSFWorkbook
Sheet	XSSFSheet	HSSFSheet	SXSSFSheet
Row	XSSFRow	HSSFRow	SXSSFRow
Cell	XSSFCell	HSSFCell	SXSSFCell

Tworzenie arkusza kalkulacyjnego w Javie - podstawy

```
Workbook workbook = new XSSFWorkbook(); // Tworzy skoroszyt typu XSSF

Sheet sheet = workbook.createSheet( s: "excel-sheet"); // Tworzy arkusz o nazwie excel-sheet

Row row = sheet.createRow( i: 0); // Tworzy wiersz na pozycji 0

row.createCell( i: 0).setCellValue("Name"); // Tworzy komórkę w kolumnie 0 w wierszu 0 i ustawia wartość "Name"

try {
    FileOutputStream out = new FileOutputStream(
            new File( pathname: "excel.xlsx")); // Tworzy plik w naszym projekcie o nazwie "excel.xlsx"
    workbook.write(out); // Zapisuje nasz skoroszyt w pliku stworzonym wyżej
    out.close(); // zamyka połączenie z plikiem
} catch (Exception e) {
    e.printStackTrace();
}
```

Tworzenie arkusza kalkulacyjnego w Javie – podstawy cd.

```
ArrayList<String[]> polacy = new ArrayList<>();
polacy.add(new String[]{"Michał", "Nowak", "76050545316"});
polacy.add(new String[]{"Szymon", "Czapla", "58111115351"});
polacy.add(new String[]{"Anna", "Mazak", "98100667793"});

Workbook workbook = new XSSFWorkbook();

Sheet sheet = workbook.createSheet( s: "excel-sheet");

Row headerRow = sheet.createRow( i: 0);

headerRow.createCell( i: 0).setCellValue("Name");
headerRow.createCell( i: 1).setCellValue("Second name");
headerRow.createCell( i: 2).setCellValue("PESEL");

for(int i = 0; i < polacy.size(); i++) {
    Row row = sheet.createRow( i: i+1);
    row.createCell( i: 0).setCellValue(polacy.get(i)[0]);
    row.createCell( i: 1).setCellValue(polacy.get(i)[1]);
    row.createCell( i: 2).setCellValue(polacy.get(i)[2]);
}
```

	A	B	C
1	Name	Second name	PESEL
2	Michał	Nowak	76050545316
3	Szymon	Czapla	58111115351
4	Anna	Mazak	98100667793
5			
6			
7			

Odczyt z arkusza kalkulacyjnego w Javie

```
try{

    FileInputStream fileInputStream = new FileInputStream( name: "excel.xlsx");
    Workbook workbook = new XSSFWorkbook(fileInputStream); // otwiera skoroszyt o podanej nazwie
    Sheet sheet = workbook.getSheetAt( i: 0); // otwiera arkusz o podanym indeksie
    Iterator<Row> rowIterator = sheet.iterator(); // tworzy iterator po wszystkich wierszach arkusza
    while (rowIterator.hasNext()) {
        Iterator<Cell> cellIterator = rowIterator.next().iterator(); // tworzy iterator po wszystkich komórkach wiersza
        while (cellIterator.hasNext()) {
            Cell cell = cellIterator.next();
            switch (cell.getCellType()) { // zwraca rodzaj komórki
                case NUMERIC: // komórka, która zawiera wartości numeryczne typu liczby całkowite, data, ułamek
                    System.out.print(cell.getNumericCellValue() + " ");
                    break;

                case STRING: // komórka, która zawiera łańcuch
                    System.out.print(cell.getStringCellValue() + " ");
                    break;
            }
        }
        System.out.println();
    }
    fileInputStream.close();

} catch (IOException e) {
    throw new RuntimeException(e);
}
```

Proste formuły w Javie

```
Workbook workbook = new XSSFWorkbook();
Sheet sheet = workbook.createSheet( s: "formulas");
Row row = sheet.createRow( i: 0);
row.createCell( i: 0).setCellValue("A");
row.createCell( i: 1).setCellValue("B");
row.createCell( i: 2).setCellValue("SUM");
row.createCell( i: 3).setCellValue("POWER");
row.createCell( i: 4).setCellValue("SQRT of first value");
row = sheet.createRow( i: 1);
row.createCell( i: 0).setCellValue(4);
row.createCell( i: 1).setCellValue(2);
Cell cell = row.createCell( i: 2);
cell.setCellFormula("SUM(A2,B2)"); // wpisujemy do komórki formułę
cell = row.createCell( i: 3);
cell.setCellFormula("POWER(A2,B2)"); // wpisujemy do komórki formułę
cell = row.createCell( i: 4);
cell.setCellFormula("SQRT(A2)"); // wpisujemy do komórki formułę

workbook.getCreationHelper().createFormulaEvaluator().evaluateAll();
// funkcja wyżej wpisuje wartości w mmiejsce wszystkich formuł

try{
    FileOutputStream out = new FileOutputStream( name: "formula.xlsx");
    workbook.write(out);
    out.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

	A	B	C	D	E
1	A	B	SUM	POWER	SQRT of first value
2	4	2	6	16	2

Apache Commons CSV

- **Czym jest Apache Commons CSV?**
- Jest to biblioteka Java należąca do rodziny Apache Commons, przeznaczona do łatwej obsługi plików CSV (Comma-Separated Values).
- Umożliwia prosty odczyt, zapis i manipulację danymi w formacie CSV.
- **Dlaczego warto jej używać?**
  - **Prosta w użyciu**: Umożliwia szybkie rozpoczęcie pracy z plikami CSV bez zbędnej konfiguracji.
  - **Elastyczność**: Obsługuje różne formaty CSV, w tym różne znaki delimitujące i niestandardowe formaty.
  - **Solidność**: To popularne rozwiązanie, które jest dobrze utrzymane i wspierane przez społeczność, co zapewnia stabilność i zaufanie użytkowników.

Co to są pliki CSV?

- **CSV (Comma-Separated Values)** to format pliku używany do przechowywania danych tabelarycznych, gdzie każda linia odpowiada jednemu rekordowi, a wartości są rozdzielane przecinkami (lub innymi separatorami, np. średnikami).

```
company,surname,forename
Foo Tech,Jones,Alice
Top Bar Hardware,Smith,Bob
Quxcorp,Garcia,Carlos
```

Instalacja Apache Commons CSV

- **Krok 1: Dodanie zależności do projektu**:
- **Krok 2: Aktualizacja zależności**:
  - Wykonaj polecenie **`mvn install`**

Apache Maven

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

Podstawowe klasy i interfejsy

- **CSVFormat**:
  - Klasa służąca do definiowania formatu pliku CSV (separator, nagłówki, cytowanie itp.).
  - Umożliwia dostosowanie opcji odczytu i zapisu.
- **CSVParser**:
  - Klasa używana do odczytu danych z pliku CSV.
  - Umożliwia iterację po rekordach i dostęp do danych w formie mapy (z nagłówkami) lub listy.
- **CSVRecord**:
  - Reprezentuje pojedynczy rekord (wiersz) w pliku CSV.
  - Umożliwia dostęp do wartości za pomocą indeksu lub nagłówka.
- **CSVPrinter**:
  - Klasa służąca do zapisywania danych do pliku CSV.
  - Umożliwia formatowanie i pisanie rekordów, w tym nagłówków.
- **Header**:
  - Interfejs umożliwiający definiowanie i zarządzanie nagłówkami plików CSV.
  - Ułatwia dostęp do danych w pliku z nagłówkami.

```
CSVParser parser = CSVFormat.EXCEL.parse(reader);
```

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

CSVFormat

Możliwość rozszerzenia formatu metodami:

```
CSVFormat.EXCEL.withNullString("N/A").withIgnoreSurroundingSpaces(true);
```

```
CSVFormat.EXCEL.withHeader("Col1", "Col2", "Col3").parse(in);
```

Wszystkie metody dostępne na stronie:
https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVFormat.html

Przykład odczytu pliku CSV

```
File csvData = new File("/path/to/csv");
CSVParser parser = CSVParser.parse(csvData, CSVFormat.RFC4180);
for (CSVRecord csvRecord : parser) {
    ...
}
```

```
Reader in = new FileReader("path/to/file.csv");
Iterable<CSVRecord> records = CSVFormat.EXCEL.parse(in);
for (CSVRecord record : records) {
    String lastName = record.get("Last Name");
    String firstName = record.get("First Name");
}
```

Również zadziała:
record.get(0)
record.get(1)

Przykład zapis pliku CSV

```
1  Writer out = new FileWriter("output.csv");
2  CSVPrinter printer = new CSVPrinter(out, CSVFormat.DEFAULT
       .withHeader("Name", "Age", "Country"));
3  printer.printRecord("Alice", "30", "Poland");
4  printer.printRecord("Bob", "25", "USA");
5  printer.flush();
6  printer.close();
7
```

```
Name,Age,Country
Alice,30,Poland
Bob,25,USA
```

Mapowanie na obiekty

```java
public class CSVToObjectExample {

    // Klasa reprezentująca dane (Person)
    public static class Person {
        private String name;
        private int age;
        private String country;
        public Person(String name, int age, String country) {
            this.name = name; this.age = age; this.country = country;
        }
        @Override
        public String toString() {
            return "Person{name='" + name + "', age=" + age + ", country='" + country + "'}";
        }
    }

    public static void main(String[] args) throws IOException {
        String filePath = "people.csv";  // Ścieżka do pliku CSV
        List<Person> people = new ArrayList<>();

        try (FileReader reader = new FileReader(filePath);
             CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT.withHeader("Name", "Age", "Country").withSkipHeaderRecord())) {

            // Mapowanie danych CSV na obiekty Person
            for (CSVRecord record : csvParser) {
                people.add(new Person(record.get("Name"), Integer.parseInt(record.get("Age")), record.get("Country")));
            }
        }

        // Wypisanie obiektów
        people.forEach(System.out::println);
    }
}
```

Obsługa różnych formatów CSV

```
//Zmiana separatora pól
CSVFormat format = CSVFormat.DEFAULT.withDelimiter(';');

//Nagłówki kolumn
CSVFormat format = CSVFormat.DEFAULT.withHeader("Imię", "Wiek","Kraj");

//Ignorowanie pustych linii
CSVFormat format = CSVFormat.DEFAULT.withIgnoreEmptyLines();

//Obsługa komentarzy
CSVFormat format = CSVFormat.DEFAULT.withCommentMarker('#');

//Znaki cudzysłowu
CSVFormat format = CSVFormat.DEFAULT.withQuote('"');

//Separator rekordów
CSVFormat format = CSVFormat.DEFAULT.withRecordSeparator("\n");

//Automatyczne przycinanie białych znaków
CSVFormat format = CSVFormat.DEFAULT.withTrim();

//Obsługa niestandardowych wartości null
CSVFormat format = CSVFormat.DEFAULT.withNullString("NULL");
```

Najczęstsze zastosowania Apache Commons CSV

Apache Commons CSV znajduje szerokie zastosowanie w różnych projektach dzięki swojej elastyczności. Oto główne obszary jego zastosowania:

- **Import i eksport danych** – umożliwia łatwą wymianę danych między systemami.
- **Analiza dużych zbiorów danych** – pozwala na filtrowanie i przetwarzanie danych, co jest przydatne w analizach.
- **Zarządzanie danymi w aplikacjach webowych** – import i eksport plików CSV ułatwia użytkownikom przenoszenie danych.
- **Automatyzacja raportów** – automatyczne generowanie raportów CSV oszczędza czas i eliminuje błędy.
- **Migracja danych między systemami**
- **Integracja z narzędziami analitycznymi** .
- **Aplikacje mobilne**

Zalety i ograniczenia Apache Commons CSV

Zalety:

- **Łatwość użycia** – Intuicyjny interfejs, szybkie wdrożenie.
- **Obsługa różnych formatów CSV** – Możliwość konfiguracji separatorów, nagłówków, znaków cudzysłowu itp.
- **Wsparcie dla dużych plików** – Obsługa strumieniowa pozwala na pracę z dużymi zbiorami danych.
- **Elastyczność konfiguracji** – Możliwość ustawienia niestandardowych opcji, np. przycinanie spacji, ignorowanie pustych linii.
- **Lekkość i wydajność** – Szybkie działanie, niska waga biblioteki.

Ograniczenia:

- **Brak wsparcia dla złożonych struktur danych** – CSV jest formatem płaskim.
- **Brak walidacji danych** – Konieczność samodzielnej kontroli poprawności.
- **Ograniczona funkcjonalność analityczna** – Wymaga dodatkowych narzędzi do zaawansowanej analizy.
- **Problemy z obsługą znaków międzynarodowych (starsze wersje)** – Niekiedy konieczna kontrola kodowania.
- **Brak automatycznego mapowania na obiekty Java** – Konieczność ręcznego mapowania danych CSV na obiekty.

Dobre praktyki w pracy z Apache Commons CSV

- Wybór odpowiedniego formatu CSV
- Walidacja danych wejściowych
- Obsługa wyjątków
- Korzystanie z nagłówków
- Zamykanie zasobów
- Używanie `flush()` i `close()`
- Optymalizacja dla dużych plików
- Przechowywanie konfiguracji formatu w jednym miejscu

Podsumowanie Apache Commons CSV

Apache Commons CSV to wydajna i elastyczna biblioteka do pracy z plikami CSV w Javie, idealna do prostych i średnio skomplikowanych zastosowań.

Oferuje łatwą obsługę odczytu i zapisu plików CSV oraz różne opcje konfiguracji, które umożliwiają dostosowanie do specyficznych wymagań. Choć ma swoje ograniczenia (np. brak zaawansowanego mapowania danych czy walidacji), jego prostota i wydajność sprawiają, że jest popularnym wyborem w wielu projektach.

Alternatywy, takie jak OpenCSV, Super CSV czy Jackson CSV, oferują dodatkowe funkcje, jak mapowanie do obiektów Java czy wsparcie dla międzynarodowych znaków, co czyni je odpowiednimi dla bardziej zaawansowanych scenariuszy.

Bibliografia

- https://blog.fileformat.com/spreadsheet/read-excel-file-in-java-with-apache-poi/
- https://medium.com/@metehankozan/creating-excel-files-in-java-with-apache-poi-0378be215b44
- https://poi.apache.org/components/spreadsheet/limitations.html
- https://poi.apache.org/components/spreadsheet/quick-guide.html
- https://chatgpt.com
- https://commons.apache.org/proper/commons-csv/user-guide.html
- https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVFormat.html
- https://commons.apache.org/proper/commons-csv/apidocs/org/apache/commons/csv/CSVParser.html