Чтение и запись файла Excel в Java с использованием Apache POI

View more categories:

Java Open source libraries

1- Что такое Apache POI?
2- Обзор Apache POI
3- Обзор Apache POI Excel
4- Библиотека Apache POI
5- Создать и записать файл Excel
6- Читать файл xsl и xslx
7- Обновить готовый файл Excel
8- Формулы и оценка
9- Применение стиля (Style)

1- Что такое Apache POI?

Apache POI это библиотека Java с открытым исходным кодо, предоставленный **Apache**, это сильная библиотека помогающая вам работать с документами **Microsoft**, как Word, Excel, Power point, Visio,...

POI это аббревиатура **"Poor Obfuscation Implementation"**. Форматы файлов Microsoft скрыты. Инженеры Apache должны были постараться чтобы понять, и они увидели что **Microsoft** создал сложные форматы, когда это не необходимо. И название библиотеки имеет происхождение от юмора.

Poor Obfuscation Implementation: Плохая реализация обфускации. (Примерный перевод).

В данной статье мы покажем вам как использовать **Apache POI** для работы с **Excel**.

2- Обзор Apache POI

Apache POI поддерживает вас при работе с форматами **Microsoft**, его классы часто имеют приставку HSSF, XSSF, HPSF, ... Смотря на приставки класса, вы можете узнать какой формат поддерживает этот класс.

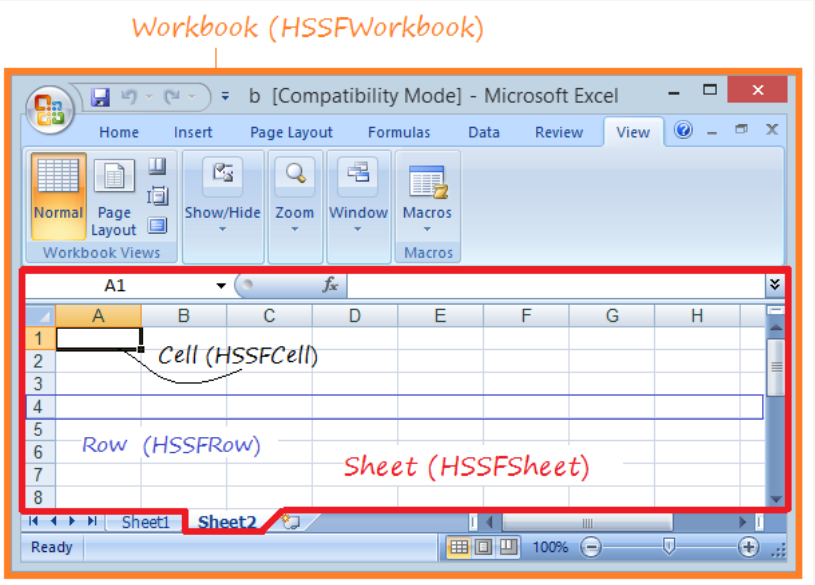
Например чтобы работать с форматом Excel (XLS) вам нужны классы:

- **HSSFWorkbook**
- **HSSFSheet**
- **HSSFCellStyle**
- **HSSFDataFormat**
- **HSSFFont**
- ...

	Prefix	Description
1	HSSF (Horrible SpreadSheet Format)	reads and writes Microsoft Excel (XLS) format files.
2	XSSF (XML SpreadSheet Format)	reads and writes Office Open XML (XLSX) format files.
3	HPSF (Horrible Property Set Format)	reads “Document Summary” information from Microsoft Office files.
4	HWPF (Horrible Word Processor Format)	aims to read and write Microsoft Word 97 (DOC) format files.
5	HSLF (Horrible Slide Layout Format)	a pure Java implementation for Microsoft PowerPoint files.
6	HDGF (Horrible DiaGram Format)	an initial pure Java implementation for Microsoft Visio binary files.
7	HPBF (Horrible PuBlisher Format)	a pure Java implementation for Microsoft Publisher files.
8	HSMF (Horrible Stupid Mail Format)	a pure Java implementation for Microsoft Outlook MSG files
9	DDF (Dreadful Drawing Format)	a package for decoding the Microsoft Office Drawing format.

3- Обзор Apache POI Excel

The image below illustrate the structure of an excel document.



Apache POI предоставляет вам интерфейсы ***Workbook***, ***Sheet***, ***Row***, ***Cell***,... и применение соответствующих классов (implementation) это **HSSFWorkbook**, **HSSFSheet**, **HSSFRow**, **HSSFCell**,...

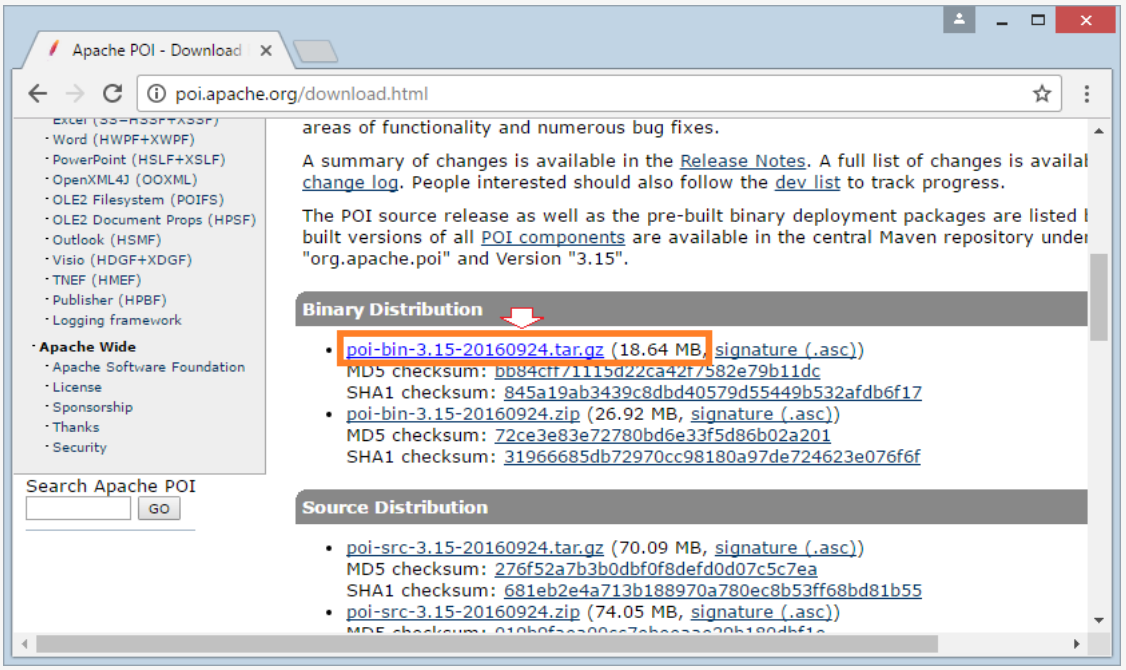
4- Библиотека Apache POI

Если ваш project использует **Maven**, вам нужно только объявить библиотеку простым способом в **pom.xml**:

```
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
<dependency>
   <groupId>org.apache.poi</groupId>
   <artifactId>poi</artifactId>
   <version>3.17</version>
</dependency>

<dependency>
   <groupId>org.apache.poi</groupId>
   <artifactId>poi-ooxml</artifactId>
   <version>3.17</version>
</dependency>
```

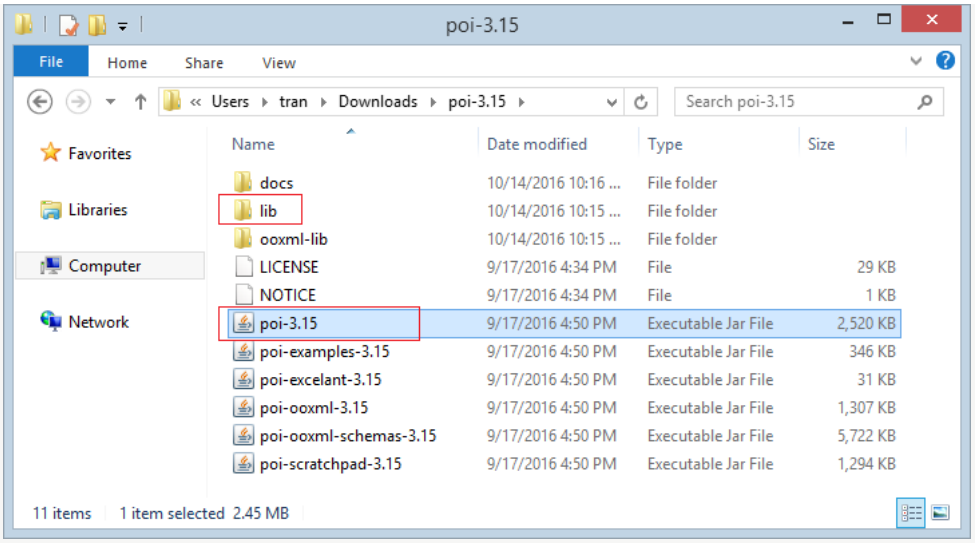
Если вы не используете Maven, то можете скачать библиотеку **Apache POI** по ссылке:

- http://poi.apache.org/download.html



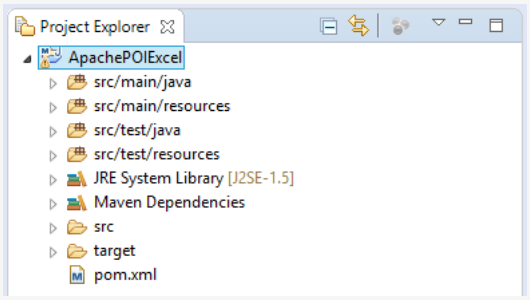
Скачать и извлечь, для работы с Excel вам нужно минимум 3 файла jar:

- poi-**.jar
- lib/commons-codec-**.jar
- lib/commons-collections4-**.jar



В данной статье, я создам простой Project Maven с названием **ApachePOIExcel**

- **Group ID:** org.o7planning
- **Artifact ID:** ApachePOIExcel



pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                        http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>org.o7planning</groupId>
    <artifactId>ApachePOIExcel</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>

        <!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>3.17</version>
        </dependency>

        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>3.17</version>
        </dependency>

    </dependencies>


</project>
```

5- Создать и записать файл Excel

В предыдущих версиях **Microsoft Office** (97-2003) файлы excel имели формат XLS и новый версии обычно используют формат XSLX. Для работы с файлами XSL вам нужно использовать классы с приставкой HSSF. Для файлов формата XSLX нужно использовать классы с приставкой XSSF.

Пример ниже является простым примером использования POI чтобы создать файл excel. Вы можете сочетать с использованием стиля (**Style**) в ячейках (Cell) чтобы создать красивый документ **Excel**. **POI Style** объясняется более детально в конце статьи.

CreateExcelDemo.java

```
package org.o7planning.apachepoiexcel.demo;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;

import org.apache.poi.hssf.usermodel.HSSFCellStyle;
import org.apache.poi.hssf.usermodel.HSSFFont;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.Row;
import org.o7planning.apachepoiexcel.model.Employee;
import org.o7planning.apachepoiexcel.model.EmployeeDAO;

public class CreateExcelDemo {
```

```
19
20      private static HSSFCellStyle createStyleForTitle(HSSFWorkbook workbook) {
21          HSSFFont font = workbook.createFont();
22          font.setBold(true);
23          HSSFCellStyle style = workbook.createCellStyle();
24          style.setFont(font);
25          return style;
26      }
27
28      public static void main(String[] args) throws IOException {
29
30          HSSFWorkbook workbook = new HSSFWorkbook();
31          HSSFSheet sheet = workbook.createSheet("Employees sheet");
32
33          List<Employee> list = EmployeeDAO.listEmployees();
34
35          int rownum = 0;
36          Cell cell;
37          Row row;
38          //
39          HSSFCellStyle style = createStyleForTitle(workbook);
40
41          row = sheet.createRow(rownum);
42
43          // EmpNo
44          cell = row.createCell(0, CellType.STRING);
45          cell.setCellValue("EmpNo");
46          cell.setCellStyle(style);
47          // EmpName
48          cell = row.createCell(1, CellType.STRING);
49          cell.setCellValue("EmpNo");
50          cell.setCellStyle(style);
51          // Salary
52          cell = row.createCell(2, CellType.STRING);
53          cell.setCellValue("Salary");
54          cell.setCellStyle(style);
55          // Grade
56          cell = row.createCell(3, CellType.STRING);
57          cell.setCellValue("Grade");
58          cell.setCellStyle(style);
59          // Bonus
60          cell = row.createCell(4, CellType.STRING);
61          cell.setCellValue("Bonus");
62          cell.setCellStyle(style);
63
64          // Data
65          for (Employee emp : list) {
66              rownum++;
67              row = sheet.createRow(rownum);
68
69              // EmpNo (A)
70              cell = row.createCell(0, CellType.STRING);
71              cell.setCellValue(emp.getEmpNo());
72              // EmpName (B)
73              cell = row.createCell(1, CellType.STRING);
74              cell.setCellValue(emp.getEmpName());
75              // Salary (C)
76              cell = row.createCell(2, CellType.NUMERIC);
77              cell.setCellValue(emp.getSalary());
78              // Grade (D)
79              cell = row.createCell(3, CellType.NUMERIC);
80              cell.setCellValue(emp.getGrade());
81              // Bonus (E)
82              String formula = "0.1*C" + (rownum + 1) + "*D" + (rownum + 1);
83              cell = row.createCell(4, CellType.FORMULA);
84              cell.setCellFormula(formula);
85          }
86          File file = new File("C:/demo/employee.xls");
87          file.getParentFile().mkdirs();
88
89          FileOutputStream outFile = new FileOutputStream(file);
90          workbook.write(outFile);
91          System.out.println("Created file: " + file.getAbsolutePath());
92
93      }
94
95  }
```

Employee.java

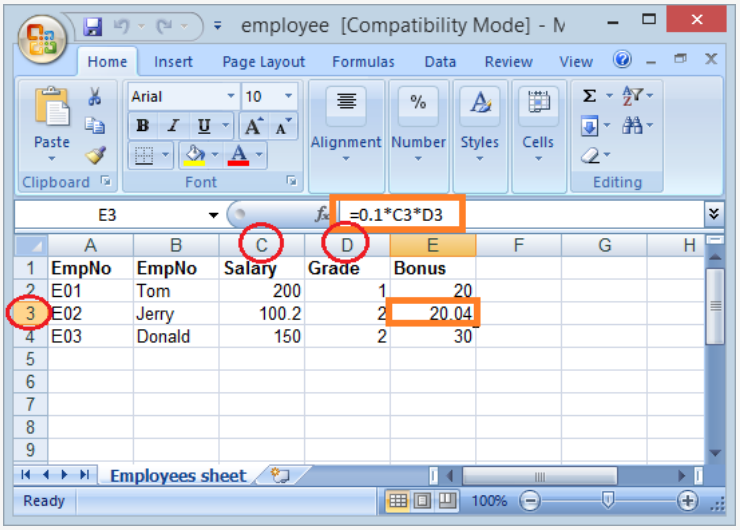
```
1   package org.o7planning.apachepoiexcel.model;
2
3   public class Employee {
4
5       private String empNo;
6       private String empName;
7
8       private Double salary;
9       private int grade;
10      private Double bonus;
11
12      public Employee(String empNo, String empName,//
13              Double salary, int grade, Double bonus) {
14          this.empNo = empNo;
15          this.empName = empName;
16          this.salary = salary;
17          this.grade = grade;
18          this.bonus = bonus;
19      }
20
21      public String getEmpNo() {
22          return empNo;
23      }
24
25      public void setEmpNo(String empNo) {
26          this.empNo = empNo;
27      }
28
29      public String getEmpName() {
30          return empName;
31      }
32
33      public void setEmpName(String empName) {
34          this.empName = empName;
35      }
36
37      public Double getSalary() {
38          return salary;
39      }
40
41      public void setSalary(Double salary) {
42          this.salary = salary;
43      }
44
45      public int getGrade() {
46          return grade;
47      }
```

```
48
49      public void setGrade(Int grade) {
50          this.grade = grade;
51      }
52
53      public Double getBonus() {
54          return bonus;
55      }
56
57      public void setBonus(Double bonus) {
58          this.bonus = bonus;
59      }
60
61  }
```

EmployeeDAO.java

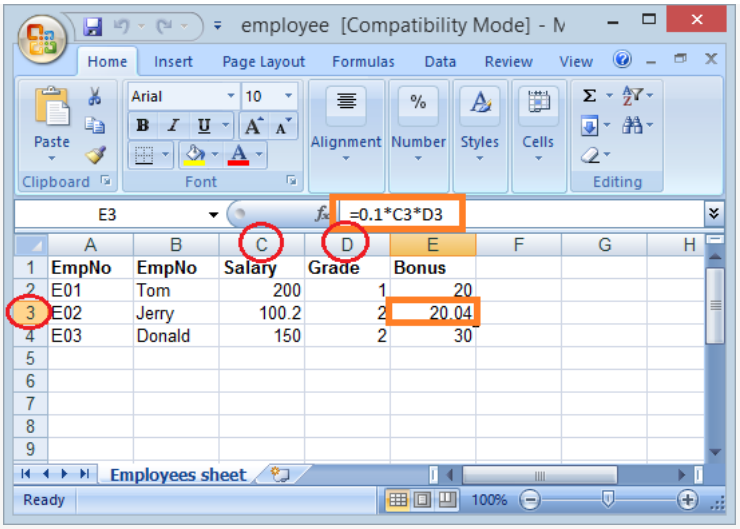
```
1   package org.o7planning.apachepoiexcel.model;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
6   public class EmployeeDAO {
7
8       public static List<Employee> listEmployees() {
9           List<Employee> list = new ArrayList<Employee>();
10
11          Employee e1 = new Employee("E01", "Tom", 200.0, 1, null);
12          Employee e2 = new Employee("E02", "Jerry", 100.2, 2, null);
13          Employee e3 = new Employee("E03", "Donald", 150.0, 2, null);
14          list.add(e1);
15          list.add(e2);
16          list.add(e3);
17          return list;
18      }
19
20  }
```

Запуск примера:



6- Читать файл xsl и xslx

Пример ниже читает простой файл excel и записывает информацию на экране **Console**. Файл excel, использующийся для чтения, это файл excel созданный в примере выше.



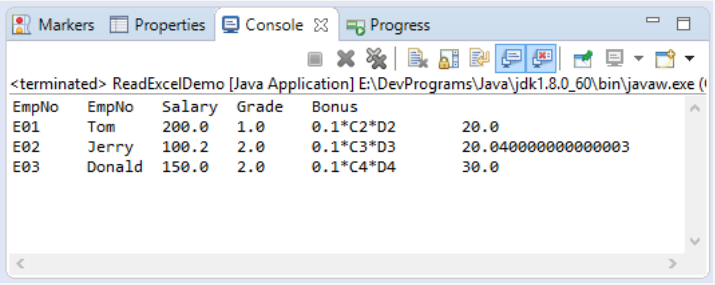
	A	B	C	D	E	F	G	H
1	**EmpNo**	**EmpNo**	**Salary**	**Grade**	**Bonus**			
2	E01	Tom	200	1	20			
3	E02	Jerry	100.2	2	20.04			
4	E03	Donald	150	2	30			
5								
6								
7								
8								
9								

> *Заметка: В данной статье я использую* ***Apache POI 3.15****, API имеет много изменений по сравнению со старой версией. Многие методы будут удалены из будущей версии (Apache POI 4.x). POI стремится использовать Enum чтобы заменить константы в его API.*

ReadExcelDemo.java

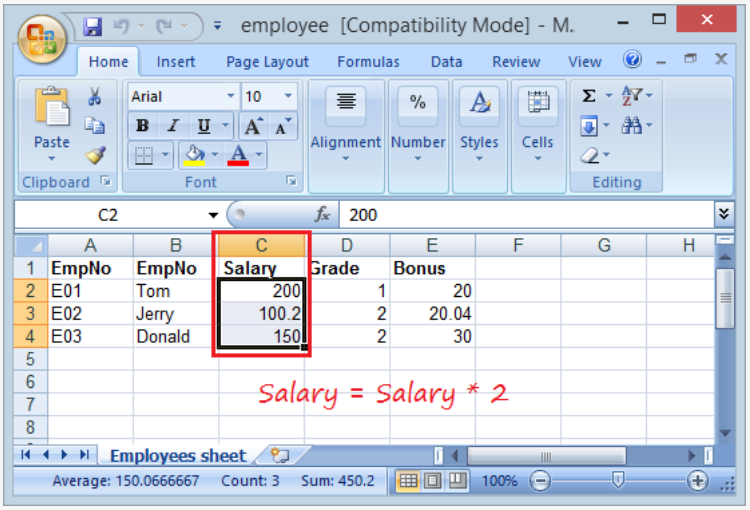
```
package org.o7planning.apachepoiexcel.demo;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Iterator;

import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.FormulaEvaluator;
import org.apache.poi.ss.usermodel.Row;

public class ReadExcelDemo {

   public static void main(String[] args) throws IOException {

      // Read XSL file
      FileInputStream inputStream = new FileInputStream(new File("C:/demo/employee.xls"));

      // Get the workbook instance for XLS file
      HSSFWorkbook workbook = new HSSFWorkbook(inputStream);

      // Get first sheet from the workbook
      HSSFSheet sheet = workbook.getSheetAt(0);

      // Get iterator to all the rows in current sheet
      Iterator<Row> rowIterator = sheet.iterator();

      while (rowIterator.hasNext()) {
         Row row = rowIterator.next();
         // Get iterator to all cells of current row
         Iterator<Cell> cellIterator = row.cellIterator();

         while (cellIterator.hasNext()) {
            Cell cell = cellIterator.next();

            // Change to getCellType() if using POI 4.x
            CellType cellType = cell.getCellTypeEnum();

            switch (cellType) {
            case _NONE:
               System.out.print("");
               System.out.print("\t");
               break;
            case BOOLEAN:
               System.out.print(cell.getBooleanCellValue());
               System.out.print("\t");
               break;
            case BLANK:
               System.out.print("");
               System.out.print("\t");
               break;
            case FORMULA:
               // Formula
               System.out.print(cell.getCellFormula());
               System.out.print("\t");

               FormulaEvaluator evaluator = workbook.getCreationHelper().createFormulaEvaluator();
               // Print out value evaluated by formula
               System.out.print(evaluator.evaluate(cell).getNumberValue());
               break;
            case NUMERIC:
               System.out.print(cell.getNumericCellValue());
               System.out.print("\t");
               break;
            case STRING:
               System.out.print(cell.getStringCellValue());
               System.out.print("\t");
               break;
            case ERROR:
               System.out.print("!");
               System.out.print("\t");
               break;
            }

         }
         System.out.println("");
      }
   }

}
```

Запуск примера:



7- Обновить готовый файл Excel

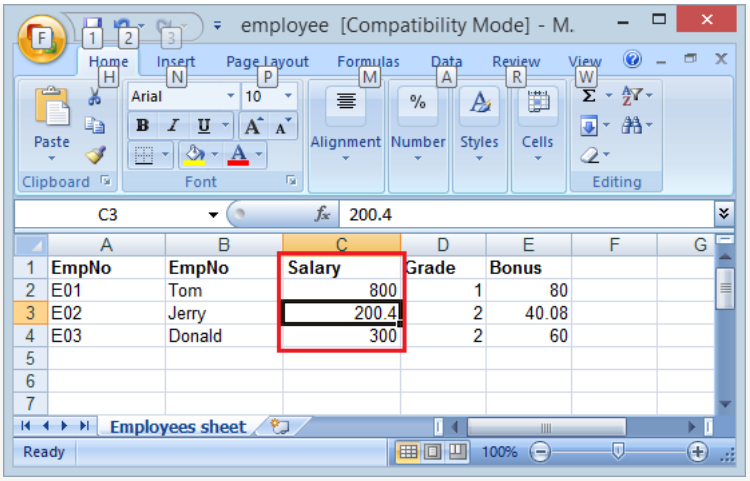
В данном примере, я читаю файл excel **employee.xls** и обновляю новые значения для столбца Salary, увеличиваю в 2 раза.



UpdateExcelDemo.java

```
package org.o7planning.apachepoiexcel.demo;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;

public class UpdateExcelDemo {

   public static void main(String[] args) throws IOException {

      File file = new File("C:/demo/employee.xls");
      // Read XSL file
      FileInputStream inputStream = new FileInputStream(file);

      // Get the workbook instance for XLS file
      HSSFWorkbook workbook = new HSSFWorkbook(inputStream);

      // Get first sheet from the workbook
      HSSFSheet sheet = workbook.getSheetAt(0);

      HSSFCell cell = sheet.getRow(1).getCell(2);
      cell.setCellValue(cell.getNumericCellValue() * 2);

      cell = sheet.getRow(2).getCell(2);
      cell.setCellValue(cell.getNumericCellValue() * 2);

      cell = sheet.getRow(3).getCell(2);
      cell.setCellValue(cell.getNumericCellValue() * 2);

      inputStream.close();

      // Write File
      FileOutputStream out = new FileOutputStream(file);
      workbook.write(out);
      out.close();

   }
}
```

Результат после обновления:



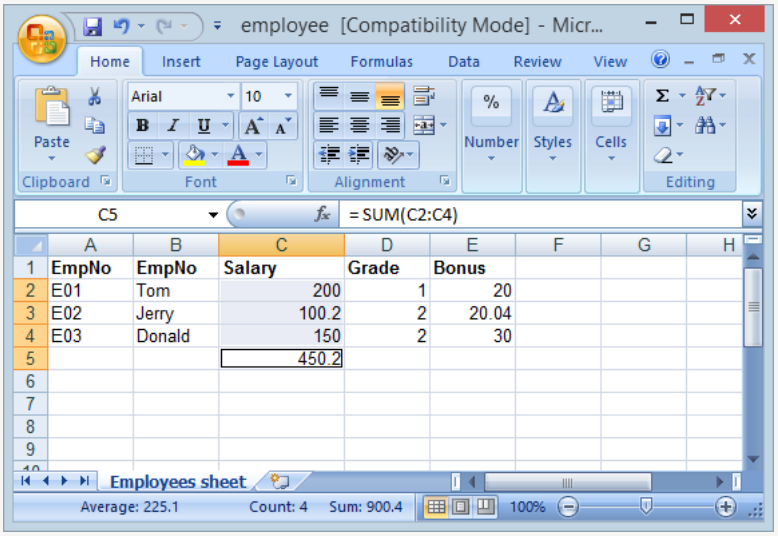
8- Формулы и оценка

Если у вас имеется знание про Excel, то вам будет легко сформулировать формулу. С **Apache POI** вы можете создать Cell вида **CellType.FORMULA**, его значение будет расчитано на основании формулы.

SUM

Например: Посчитать сумму ячеек одного столбца "C" начиная со 2-ой строки до 4-ой:

```
// Create Cell type of FORMULA
cell = row.createCell(rowIndex, CellType.FORMULA);

// Set formula
cell.setCellFormula("SUM(C2:C4)");
```



Пример формулы:

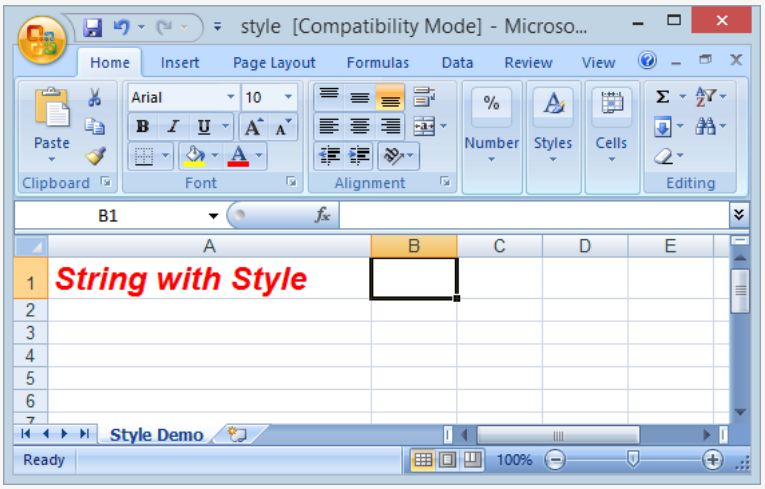
```
cell = row.createCell(rowIndex, CellType.FORMULA);
cell.setCellFormula("0.1*C2*D3");
```

Для ячейки вида **FORMULA**, вы можете распечатать ее формулу и использовать **FormulaEvaluator**, чтобы посчитать значение ячейки данная формулой.

```
// Formula
String formula = cell.getCellFormula();

FormulaEvaluator evaluator
    = workbook.getCreationHelper().createFormulaEvaluator();

// CellValue
CellValue cellValue = evaluator.evaluate(cell);

double value = cellValue.getNumberValue();
String value = cellValue.getStringValue();
boolean value = cellValue.getBooleanValue();
// ...
```

9- Применение стиля (Style)

Пример:



StyleDemo.java

```java
package org.o7planning.apachepoiexcel.demo;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFCellStyle;
import org.apache.poi.hssf.usermodel.HSSFFont;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.IndexedColors;

public class StyleDemo {

    private static HSSFCellStyle getSampleStyle(HSSFWorkbook workbook) {
        // Font
        HSSFFont font = workbook.createFont();
        font.setBold(true);
        font.setItalic(true);

        // Font Height
        font.setFontHeightInPoints((short) 18);

        // Font Color
        font.setColor(IndexedColors.RED.index);

        // Style
        HSSFCellStyle style = workbook.createCellStyle();
        style.setFont(font);

        return style;
    }

    public static void main(String[] args) throws IOException {

        HSSFWorkbook workbook = new HSSFWorkbook();
        HSSFSheet sheet = workbook.createSheet("Style Demo");

        HSSFRow row = sheet.createRow(0);

        //
        HSSFCell cell = row.createCell(0);
        cell.setCellValue("String with Style");

        HSSFCellStyle style = getSampleStyle(workbook);
        cell.setCellStyle(style);

        File file = new File("C:/demo/style.xls");
        file.getParentFile().mkdirs();

        FileOutputStream outFile = new FileOutputStream(file);
        workbook.write(outFile);
        System.out.println("Created file: " + file.getAbsolutePath());

    }

}
```


View more categories:

Java Open source libraries

Java Open source libraries

- Получение информации об оборудовании в приложении Java
- Руководство Java Json Processing API (JSONP)
- Использование Scribe OAuth Java API с Google OAuth 2
- Руководство Restfb Java Facebook Graph API
- Руководство Java JDOM2
- Руководство Java XStream
- Использование Java JSoup для анализа кода HTML
- Извлечение географической информации на основе IP-адреса с использованием GeoIP2 Java API
- Чтение и запись файла Excel в Java с использованием Apache POI

Самые новые руководства

- Создать бесплатный VPS в Google Cloud
- Загрузить SAP Trial System
- Пример Upload file с Spring Boot и AngularJS
- Защита Spring Boot RESTful Service используя Auth0 JWT
- Войти в систему используя социальную сеть с OAuth2 в Spring Boot
- Руководство Spring Boot и Mustache
- Руководство Spring Boot и Groovy
- Пример Upload file с Spring Boot и jQuery Ajax
- Разбиение по страницам (Pagination) в Java Hibernate
- Пример CRUD с Spring Boot, REST и AngularJS