МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
Национальный исследовательский ядерный университет «МИФИ»

Институт
интеллектуальных кибернетических систем

Кафедра №22 «Кибернетика»

Направление подготовки 09.03.04 Программная инженерия

Пояснительная записка

к учебно-исследовательской работе студента на тему:

Интеграция электронных таблиц MS Excel в Java.

Группа Б15-503

Студент _____ (подпись) Эргашев О.И. (ФИО)

Руководитель _____ (подпись) Цыганов А. А. (ФИО)

Научный консультант _____ (подпись) _____ (ФИО)

Оценка руководителя _____ (0-15 баллов) Оценка консультанта _____ (0-15 баллов)

Итоговая оценка _____ (0-100 баллов) ECTS _____

Комиссия

Председатель _____ (подпись) _____ (ФИО)

_____ (подпись) _____ (ФИО)

_____ (подпись) _____ (ФИО)

_____ (подпись) _____ (ФИО)

Москва 2018

Введение

Описание:

В современном мире очень много случаев, при которых необходимо интегрировать MS Excel с Java. Например, при разработке Enterprise-приложения в некой финансовой сфере, вам необходимо предоставить счет для заинтересованных лиц, а проще всего выставлять счет на MS Excel.

Обзор существующих API MS Excel для Java:

Рассмотрим основные API[1]:

-- Docx4j - это API с открытым исходным кодом, для создания и манипулирования документами формата Microsoft Open XML, к которым отросятся Word docx, Powerpoint pptx, Excel xlsx файлы. Он очень похож на Microsoft OpenXML SDK, но реализован на языке Java. Docx4j использует JAXB архитектуру для создания представления объекта в памяти. Docx4j акцентирует свое внимание на всесторонней поддержке заявленного формата, но от пользователя данного API требуется знание и понимание технологии JAXB и структуры Open XML.

-- Apache POI - это набор API с открытым исходным кодом, который предлагает определенные функции для чтения и записи различных документов, базирующихся на Office Open XML стандартах (OOXML) и Microsoft OLE2 формате документов (OLE2). OLE2 файлы включают большинство Microsoft Office форматов, таких как doc, xls, ppt. Office Open XML формат это новый стандарт базирующийся на XML разметке, и используется в файлах Microsoft office 2007 и старше.

-- Aspose for Java - набор платных Java APIs, которые помогают разработчикам в работе с популярными форматами бизнес файлов, такими как документы Microsoft Word, таблицы Microsoft Excel, презентации Microsoft PowerPoint, PDF файлы Adobe Acrobat, emails, изображения, штрих-коды и оптические распознавания символов.

Каждое API проектируется для того, чтобы выполнять широкий спектр создания документов, различные манипуляции и преобразования быстро и легко, экономя время и позволяя разработчикам успешно программировать. Ни один API с открытым исходным кодом не имеет одной и той же комплексной поддержки функций.

Все Aspose’s APIs используют простую объектную модель документа, а одно API предназначено для работы с набором связанных форматов. Aspose’s Microsoft Office APIs, Aspose.Cells,

Aspose.Words, Aspose.Slides, Aspose.Email, и Aspose.Tasks легки в работе, эффективны, надежны и независимы от других библиотек.

Преимуществом APIs с открытым исходным кодом является то, что они бесплатны и каждый может настроить их под свои задачи и цели. Это очень удобно, если у пользователя есть достаточно времени и ресурсов. Однако данные APIs не всегда имеют поддержку или документацию, и поддерживают небольшое количество функций и вариантов. Этот недостаток стоит разработчикам времени, и сокращает надежность их приложений. К преимуществам проприетарных (коммерческих) API можно отнести комплексную поддержку функционала с подробной документацией, регулярное обновление, гарантию отсутствия ошибок и обратную связь с разработчиками APIs.

В данной программе будем использовать Apache POI

Содержание

1. Компоненты Apache POI

1.1 Описание компонетов

Horrible Spreadsheet Format	Компонент чтения и записи файлов MS-Excel, формат XLS
XML Spreadsheet Format Компонент	Чтения и записи файлов MS-Excel, формат XLSX
Horrible Property Set Format	Компонент получения наборов свойств файлов MS-Office
Horrible Word Processor Format	Компонент чтения и записи файлов MS-Word, формат DOC
XML Word Processor Format	Компонент чтения и записи файлов MS-Word, формат DOCX
Horrible Slide Layout Format	Компонент чтения и записи файлов PowerPoint, формат PPT
XML Slide Layout Format	Компонент чтения и записи файлов PowerPoint, формат PPTX
Horrible DiaGram Format	Компонент работы с файлами MS-Visio, формат VSD
XML DiaGram Format	Компонент работы с файлами MS-Visio, формат VSDX

1.2 Список компонентов

Таблица 1.1 – Список компонентов[2]

Наименование(артефакт)	**Необходимые компоненты**
poi	commons-logging, commons-codec, commons-collections, log4j
poi-scratchpad	poi
poi-ooxml	poi, poi-ooxml-schemas
poi-ooxml-schemas	xmlbeans
poi-examples	poi, poi-scratchpad, poi-ooxml
ooxml-schemas	xmlbeans
ooxml-security	xmlbeans

В данной работе рассматриваются следующие классы, используемые для работы с файлами Excel из приложений Java.

-- рабочая книга - HSSFWorkbook, XSSFWorkbook

-- лист книги - HSSFSheet, XSSFSheet

-- строка - HSSFRow, XSSFRow

-- ячейка - HSSFCell, XSSFCell

-- стиль - стили ячеек HSSFCellStyle, XSSFCellStyle

-- шрифт - шрифт ячеек HSSFFont, XSSFFont

Поскольку описание всех классов и методов не разместить на одной странице, то можно ее прочитать в офф док.

2. Классы и методы Apache POI для работы с файлами Excel

2.1 Рабочая книга HSSFWorkbook, XSSFWorkbook

-- HSSFWorkbook org.apache.poi.hssf.usermodel класс чтения и записи файлов Microsoft Excel в формате .xls, совместим с версиями MS-Office 97-2003;

-- XSSFWorkbook org.apache.poi.xssf.usermodel класс чтения и записи файлов Microsoft Excel в формате .xlsx, совместим с MS-Office 2007 или более поздней версии.

2.1.1 Конструкторы класса HSSFWorkbook

```
HSSFWorkbook ();
HSSFWorkbook (InternalWorkbook book);
HSSFWorkbook (POIFSFileSystem  fs);
HSSFWorkbook (NPOIFSFileSystem fs);
HSSFWorkbook (POIFSFileSystem  fs, boolean preserveNodes);
HSSFWorkbook (DirectoryNode directory,
POIFSFileSystem fs,
boolean preserveNodes);
HSSFWorkbook (DirectoryNode directory, boolean preserveNodes);
HSSFWorkbook (InputStream s);
HSSFWorkbook (InputStream s, boolean preserveNodes);
```

preservenodes является необязательным параметром, который определяет необходимость сохранения узлов типа макросы.

2.1.2 Конструкторы класса XSSFWorkbook

```
XSSFWorkbook ();
// workbookType  создать .xlsx или .xlsm
XSSFWorkbook (XSSFWorkbookType workbookType);
XSSFWorkbook (OPCPackage pkg );
```

```
XSSFWorkbook (InputStream is);
XSSFWorkbook (File file);
XSSFWorkbook (String path);
```

2.1.3 Основные методы HSSFWorkbook, XSSFWorkbook

Таблица 2.1 – Основные методы HSSFWorkbook, XSSFWorkbook

Метод	**Описание**
createSheet()	Создание страницы книги HSSFSheet, XSSFSheet
createSheet(String name)	Создание страницы с определенным наименованием
createFont()	Создание шрифта
createCellStyle()	Создание стиля

С полным перечнем всех методов класса XSSFWorkbook можно познакомиться на странице http://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFWorkbook.html.

2.2 Классы листов книги, HSSFSheet, XSSFSheet

-- org.apache.poi.hssf.usermodel.**HSSFSheet**

-- org.apache.poi.xssf.usermodel.**XSSFSheet**

Классы HSSFSheet, XSSFSheet включают свойства и методы создания строк, определения размера колонок, слияния ячеек в одну область и т.д.

2.2.1 Основные методы классов работы с листами

Таблица 2.2 – Основные методы классов работы с листами

Метод	**Описание**
addMergedRegion(CellRangeAddress)	Определение области слияния ячеек страницы
autoSizeColumn(int column)	Автоматическая настройка ширины колонки column (отсчет от
setColumnWidth(int column, int width)	Настройка ширины колонки column (отсчет от 0)
createRow(int row)	Создание строки row (отсчет от 0)
getRow(int row)	Получение ссылки на строку row (отсчет от 0)

С полным перечнем всех методов класса XSSFSheet можно познакомиться на странице https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFSheet.html.

2.3 Классы строк HSSFRow, XSSFRow

-- org.apache.poi.hssf.usermodel.**HSSFRow**

-- org.apache.poi.xssf.usermodel.**XSSFRow**

Классы HSSFRow, XSSFRow включают свойства и методы работы со строками, создания ячеек в строке и т.д.

2.3.1 Основные методы классов HSSFRow, XSSFRow

Таблица 2.3 – Основные методы классов HSSFRow, XSSFRow

Метод	**Описание**
setHeight(short)	Определение высоты строки
getHeight()	Получение значения высоты в twips'ax (1/20)
getHeightInPoints()	Получение значение высоты
createCell (int)	Создание ячейки в строке (отсчет от 0)
getCell(int)	Получение ссылки на ячейку
getFirstCellNum()	Получение номера первой ячейки в строке
setRowStyle(CellStyle)	Определение стиля всей строки

С полным перечнем всех методов класса XSSFRow можно познакомиться на странице http://poi.apach

2.4 Классы ячеек HSSFCell, XSSFCell

Ячейки электронной таблицы используются для размещения информации. В ячейке может быть представлено числовое значение, текст или формула. Также ячейка может содержать комментарий.

Классы HSSFCell, XSSFCell включают свойства и методы работы с ячейками таблицы.

2.4.1 Основные методы классов HSSFCell, XSSFCell

- org.apache.poi.hssf.usermodel.**HSSFCell** - org.apache.poi.xssf.usermodel.**XSSFCell**

С полным перечнем всех методов класса XSSFCell можно познакомиться на странице http://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFCell.html

Таблица 2.4 – Основные методы классов HSSFRow, XSSFRow

Метод	**Описание**
getBooleanCellValue()	Чтение логического значения ячейки
getDateCellValue()	Чтение значения ячейки типа java.util.Date
getNumericCellValue()	Чтение числового значения ячейки типа double
getStringCellValue()	Чтение текстового значения ячейки (java.lang.String)
setCellValue(boolean)	Опреаделение логического значения ячейки
setCellValue(java.util.Calendar)	Определение значения ячейки типа даты
setCellValue(java.util.Date)	Определение значения ячейки типа даты
getCellTypeEnum()	Чтение типа значения ячейки CellType
setCellComment(Comment)	Запись комментария в ячейку
getCellComment()	Чтение комментария ячейки
removeCellComment()	Удаление комментария ячейки
setHyperlink(Hyperlink)	Запись гиперссылки в ячейку
getHyperlink()	Чтение гиперссылки XSSFHyperlink в ячейке
removeHyperlink()	Удаления гиперссылки ячейки
getCellFormula()	Чтение формулы, например SUM(C4:E4)
setCellFormula(String)	Определение формулы, например =SUM(C4:E4)
getCellStyle()	Чтение стиля ячейки (XSSFCellStyle)
setCellStyle(CellStyle)	Определение стиля ячейки
getColumnIndex()	Определение индекса ячейки
setAsActiveCell()	Определение активности ячейки

2.5 Классы стилей ячеек HSSFCellStyle, XSSFCellStyle

С полным перечнем всех свойств и методов класса XSSFCellStyle можно познакомиться на странице

http://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/CellStyle.html

Ниже в качестве примера представлен метод, формирующий стиль ячейки, в которой :

- текст центрируется по вертикали и горизонтали; - обрамление ячейки представляет тонкую черную линию по периметру; - текст переносится на следующую строку (не ячейку), если не вмещается в размер ячейки.

```
private XSSFCellStyle createCellStyle(XSSFWorkbook book) {
  BorderStyle thin  = BorderStyle.THIN;
  short       black = IndexedColors.BLACK.getIndex();

  XSSFCellStyle style = book.createCellStyle();

  style.setWrapText(true);
  style.setAlignment        (HorizontalAlignment.CENTER);
  style.setVerticalAlignment(VerticalAlignment  .CENTER);

  style.setBorderTop        (thin);
  style.setBorderBottom     (thin);
  style.setBorderRight      (thin);
  style.setBorderLeft       (thin);

  style.setTopBorderColor   (black);
  style.setRightBorderColor (black);
  style.setBottomBorderColor(black);
  style.setLeftBorderColor  (black);

  return style;
}
```

Метод setWrapText позволяет определить флаг переноса текста в ячейке согласно ее размеру (ширине). Чтобы перенести текст принудительно, можно в текстовой строке установить символы CRCL, например "Разделитель \r \n текста".

2.6 Классы шрифтов HSSFFont, XSSFFont

С полным перечнем всех свойств и методов класса XSSFFont можно познакомиться на странице

http://poi.apache.org/apidocs/org/apache/poi/ss/usermodel/Font.html

Ниже в качестве примера представлен метод, формирующий шрифт типа "Times New Roman" :

```
private XSSFFont createCellFont(XSSFWorkbook book)  {
```

```
  XSSFFont font = workBook.createFont();
  font.setFontHeightInPoints((short)  12);
  font.setBoldweight(XSSFFont.BOLDWEIGHT_BOLD);
  font.setFontName("Times New Roman");
  return(font);
}
.  .  .
HSSFCellStyle style = book.createCellStyle(); style.setFont(createCellFont(book)
```

3. Инструкция

-- Для обращения к MS Excel версии до 2003 включительно года с Java используется класс HSSFWorkbook

-- Для обращения к MS Excel версии 2007 и позднее с Java используется класс XSSFWorkbook

-- При операциях **Обновление** или **Запись** необходимо, чтобы MS Excel был закрыт.

3.1 Чтение ячейки с MS Excel

Чтобы считать данные с **file.xlsx** необходимо исполнить следующие шаги:

```
//filePath – это путь до MS Excel
Workbook book = new XSSFWorkbook(new FileInputStream(filePath);
//считывается лист по индексу sheet_index. sheet_index начинается с 0
Sheet sheet = book.getSheetAt(sheet_index);
//считывается row по индексу row_index. row_index начинается с 0
Row row = sheet.getRow(row_index);
//считывается cell по индексу cell_index. cell_index начинается с 0
Cell cell = sheet.getCell(cell_index);
```

3.2 Запись в ячейку MS Excel

```
Workbook book = new XSSFWorkbook();
//name – имя листа
Sheet sheet = book.createSheet(name);
Row row = sheet.createRow(i);
Cell cell = row.createCell(j);
FileInputStream fileOut = new FileInputStream(filePath);
book.write(fileOut);
fileOut.close();
```

3.3 Обновление ячейки в существующем листе MS Excel

```
Workbook workbook = new XSSFWorkbook(new FileInputStream(filePath));
Sheet sheet = workbook.getSheetAt(i);
Row row = sheet.getRow(j);
Cell cell = row.getCell(k);
cell.setCellValue(value);
```

3.4 Подготовка: загрузка библиотек и зависимостей

Чтобы использовать apache POI, вам нужно скачать jar файлы и добавить их через Intellij IDEA вручную, или вы можете предоставить это Maven.

Во втором случае вам нужно просто добавить следующие две зависимости в pom.xml:

```
<dependencies>
  <dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.12</version>
  </dependency>
  <dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>3.12</version>
  </dependency>
</dependencies>
```

Самое удобное в Maven — что он загрузит не только указанные poi.jar и poi-ooxml.jar, но и все jar файлы, которые используются внутри, то есть xmlbeans-2.6.0.jar, stax-api-1.0.1.jar, poi-ooxml-schemas-3.12.jar и commons-codec-1.9.jar

4. Выполнение

1. Создать проект на java с помощью maven.
2. Следовать инструкции "Подготовка:..." описанная выше.
3. Создать Excel файл в корневой папке проекта.
4. Записать в A1 и A2 любые целые числа.
5. Сохранить Excel файл.
6. **Закрыть** Excel файл. (ОБЯЗЕТЬНОЕ УСЛОВИЕ, т.к. apache POI API может работать только с закрытым фалом)
7. В папке src/main/java создать класс IOCell

   7.1. Создать поле

```
        private File filePath
```

   7.2. Создать конструктор

```
        IOCell(String filePath) { this.filePath = new File(filePath)}
```

   7.3. Создать метод для чтения с Excel в Java

```
public void setCell(int row, int column, double val) {
  Workbook workbook = null;
  try (FileInputStream file = new FileInputStream(filePath)) {
    workbook = new XSSFWorkbook(file);
    Sheet sheet = workbook.getSheetAt(0);
       sheet.getRow(row).getCell(column).setCellValue(val);
  } catch (IOException e) {
    e.printStackTrace();
  }
  try (OutputStream fileOut = new FileOutputStream(filePath)) {
       workbook.write(fileOu;
  } catch (FileNotFoundException e) {
    System.out.println("file is not exist AAAA");
  } catch (IOException e) {
    e.printStackTrace();
```

```
    }
}
```

7.4. В папке src/main/java создать класс Main

7.4.1. Создать поле

```
private static final String filePath = "NAME_OF_EXCEL_FILE";
```

7.4.2. Создать метод

```
public static void main(String[] args) {
    IOCell ioCell = new IOCell(filePath);
    Cell x = ioCell.getCell(0, 1, 0);
    Cell y = ioCell.getCell(0,  1, 1);
    System.out.println("first number: " + x.toString());
       System.out.println("second number: " + y.toString());
    //Write x * y
    ioCell.setCell(4, 0, x.getNumericCellValue() *
    y.getNumericCellValue());
    //Write x + y
    ioCell.setCell(4, 1, x.getNumericCellValue() +
    y.getNumericCellValue());
    System.out.println("Interactions is complete successfully");
}
```

7.5. Запускаем приложение и смотрим в консоль.

Литература

1. Выбор API для работы с файлами Microsoft Office на языке Java - https://poi.apache.org/apidocs/index.html(дата обращения: 2018-11-15).
2. Официальная документация - https://poi.apache.org/apidocs/index.html(дата обращения: 2018-11-15).