# POI与Easyexcel的优缺点

## POI

### 优点

1、Apache官方维护，生态成熟稳定

2、功能全面，支持Excel的各种复杂操作（公式、图表、样式等）

3、文档完善，社区资源丰富

4、支持.xls和.xlsx格式

### 缺点

1、处理大文件时内存占用高，容易OOM

2、API相对复杂，学习成本较高

3、性能一般，大数据量处理较慢

## EasyExcel

### 优点

1、阿里开源，专门针对大文件优化

2、内存占用极低，能处理几十万行数据不OOM

3、API简洁易用，上手快

4、读写性能优秀

5、支持注解方式，代码更简洁

### 缺点

1、功能相对简单，复杂Excel操作支持有限

2、主要专注于数据读写，样式、公式等高级功能较弱

3、社区相对较小

## 选择建议

1、大数据量处理：选EasyExcel，内存友好

2、复杂Excel操作：选POI，功能更全面

3、简单读写需求：选EasyExcel，开发效率高

4、企业级应用：POI更稳妥，EasyExcel更高效

# 将数据库的数据导出到excel中

开发思路

https://blog.csdn.net/u011900448/article/details/53097382

引入的poi的jar包

Poi.jar

poi-ooxml.jar

poi-ooxml-schemas.jar

# 依赖

## poi-ooxm

<dependency>  
<groupId>org.apache.poi</groupId>  
<artifactId>poi-ooxml</artifactId>  
<version>4.1.2</version>  
</dependency>

Apache POI-OOXML是Java库，用于处理Microsoft Office的Open XML格式文档（XLSX、DOCX、PPTX等）。这个依赖的主要作用和功能包括：

1、Excel文件处理：

创建、读取、修改XLSX格式的Excel文件

支持公式、样式、图表等高级功能

处理大型电子表格的流式API

2、Word文档处理：

操作DOCX格式的Word文档

添加/修改文本、表格、图片等元素

设置文档格式和样式

3、PowerPoint演示文稿：

创建和编辑PPTX格式的演示文稿

管理幻灯片、图形和动画

4、XML处理能力：

基于Office Open XML标准

内部使用XML技术处理文档结构

5、常见使用场景：

报表生成和数据导出到Excel

文档自动化生成

解析和提取Office文档中的数据

批量处理Office文件

相比旧版POI（poi依赖），poi-ooxml专门处理较新的Office格式（2007+），支持更多功能，并提供更好的大文件处理性能。

# HSSFWorkbook与XSSFWorkbook的区别

HSSFWorkbook:是操作Excel2003以前（包括2003）的版本，扩展名是.xls，XSSFWorkbook:是操作Excel2007的版本，扩展名是.xlsx，对于不同版本的EXCEL文档要使用不同的工具类，如果使用错了，会提示如下错误信息。org.apache.poi.openxml4j.exceptions.InvalidOperationException

org.apache.poi.poifs.filesystem.OfficeXmlFileException

# Poi cellType类型

CELL\_TYPE\_NUMERIC 数值型 0

CELL\_TYPE\_STRING 字符串型 1

CELL\_TYPE\_FORMULA 公式型 2

CELL\_TYPE\_BLANK 空值 3

CELL\_TYPE\_BOOLEAN 布尔型 4

CELL\_TYPE\_ERROR 错误 5

# 学习思路

（1）创建新的工作簿

（2）创建新的Sheet页

（3）创建单元格

（4）创建一个时间格式的单元格

（5）处理不同内容格式的单元格

（6）遍历工作簿的行和列并获取单元格内容。

（7）文本提取

说明：poi版本是3.11

## 创建新的工作薄(excel)

Workbook wb = new HSSFWorkbook();//定义一个新的工作簿

FileOutputStream fos = new FileOutputStream("地址");//文件输出流

wb.write(fos);

fos.close();

## 创建Sheet页

Workbook wb = new HSSFWorkbook();//定义一个新的工作簿

wb.createSheet("名字");//创建sheet页

FileOutputStream fos = new FileOutputStream("地址");

wb.write(fos);

fos.close();

## 创建单元格

Workbook wb = new HSSFWorkbook();//定义一个新的工作簿

Sheet sheet = wb.createSheet("名字");

Row row = sheet.createRow(0); //创建一行

Cell cell = row.createCell(0);//创建一个单元格

cell.setCellValue(); //给单元格设置值

或者

row.createCell(0).setCellValue();

## 创建日期时间单元格

Workbook wb = new HSSFWorkbook();//定义一个新的工作簿

Sheet sheet = wb.createSheet("名字");//创建一个Sheet页

Row row = sheet.createRow(0); //创建一行

Cell cell = row.createCell(0);//创建一个单元格

CreationHelper ch = wb.getCreationHelper();

CellStyle cellstyle= wb.createCellStyle(); //创建单元格样式

cellstyle.setDateFormat(ch.createDataFormat().getFormat(yyyy-mm-dd hh:MM:ss));

cell.setCellValue(new Date());//设置单元格日期

cell.setCellStyle(cellstyle);//设置单元格样式

或者

cell.setCellValue(Calendar.getInstance());

cell.setCellStyle(cellstyle);

## 单元格中可以设置各种类型的值

//设置单元格的值

row.createCell(0).setCellValue();

## 遍历工作簿的行和列并获取单元格内容

try{

InputStream is = new FileInputStream("D:\\1.xls");//读取工作簿

POIFSFileSystem fs = new POIFSFileSystem(is);

HSSFWorkbook hssf = new HSSFWorkbook(fs);

HSSFSheet sheet = hssf.getSheetAt(0);//获取第一个sheet页

if(sheet!=null){

//遍历行

for(int i=0;i<=sheet.getLastRowNum();i++){

HSSFRow hs = sheet.getRow(i);//得到行

//遍历列

for(int j=0;j<=hs.getLastCellNum();j++){

HSSFCell hss = hs.getCell(j);//得到列

if(hss==null){

continue;

}

System.out.print(" "+getCellValue(hss));

}

System.out.println();

}

}

}catch (Exception e) {

e.printStackTrace();

}

}

//读取单元格的值

private static String getCellValue(HSSFCell hs){

//单元格的类型是数值

if(hs.getCellType()==HSSFCell.CELL\_TYPE\_NUMERIC){

return String.valueOf(hs.getNumericCellValue());

}else{

return hs.getStringCellValue();

}

}

## 获取execl文本内容

InputStream is = new FileInputStream("文件地址")；

POIFSFileSystem fs = new POIFSystem(is);

HSSFWorkbook wb = new HSSFWorkbook(fs);

ExcelExtractor ex = new ExcelExtractor(wb);

ex.setIncludeSheetNames(false);//我们不需要sheet的名字

System.out.println(ex.getText());

## 设置字体

Workbook wb = new HSSFWorkbook();//创建工作簿

Sheet sheet = wb.createSheet("我的工作簿");

Row row = sheet.createRow(0);

//创建一个字体类

Font font = wb.createFont();

font.setFontHeightInPoints((short)24);

font.setColor((short)220);

font.setItalic(true);

font.setStrikeout(true);

CellStyle cellstyle = wb.createCellStyle();

cellstyle.setFont(font);

Cell cell = row.createCell(0);

cell.setCellValue("靳世辉");

cell.setCellStyle(cellstyle);

FileOutputStream fos = new FileOutputStream("D:\\2.xls");

wb.write(fos);

fos.close();

## 读取和重写工作簿

InputStream in = new FileInputStream("D:\\2.xls");

POIFSFileSystem ps = new POIFSFileSystem(in);

Workbook wb = new HSSFWorkbook(ps);

Sheet sheet = wb.getSheetAt(0);

Row row = sheet.getRow(0);

Cell cell = row.getCell(0);

if(cell==null){

cell = row.createCell(5);

}

cell.setCellType(Cell.CELL\_TYPE\_STRING);

cell.setCellValue("中国");

FileOutputStream fos = new FileOutputStream("D:\\2.xls");

wb.write(fos);

ExcelExtractor ex = new ExcelExtractor((HSSFWorkbook) wb);

ex.setIncludeSheetNames(false);//我们不需要sheet的名字

System.out.println(ex.getText());

fos.close();

## 单元格换行

Workbook wb = new HSSFWorkbook();//定义新的工作簿

Sheet sheet = wb.createSheet("1");

Row row = sheet.createRow(0);

Cell cell = row.createCell(0);

cell.setCellValue("我要换行，可以吗？");

CellStyle cs = wb.createCellStyle();

cs.setWrapText(true);//设置换行

cell.setCellStyle(cs);

//设置行高

row.setHeightInPoints(2\*sheet.getDefaultRowHeightInPoints());

//设置单元格宽度

sheet.autoSizeColumn(2);

FileOutputStream fos = new FileOutputStream("D:\\2.xls");

wb.write(fos);

fos.close();