**POI**

## 一、解析Excel工具

**1 Apache POI是一种流行的API**，它允许程序员使用Java程序创建，修改和显示MS Office文件。这由Apache软件基金会开发使用Java分布式设计或修改Microsoft Office文件的开源库。它包含类和方法对用户输入数据或文件到MS Office文档进行解码。

**2 工具版本**

**HSSFWorkBook：**操作2003版本以前的（包括2003版本），扩展名.xls，该类在org.apache.poi:poi中

**XSSFWorkBook：**操作2007版本以后的（包括2007版本），拓展名.xlsx，该类在**org.apache.poi:**poi-ooxml中

**SXSSFWorkBook：**对于海量的数据进行操作

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

**3 API组件**

POIFS (较差混淆技术实现文件系统) : 此组件是所有其他POI元件的基本因素。它被用来明确地读取不同的文件。

HSSF (可怕的电子表格格式) : 它被用来读取和写入MS-Excel文件的xls格式。

XSSF (XML格式) : 它是用于MS-Excel中XLSX文件格式。

HPSF (可怕的属性设置格式) : 它用来提取MS-Office文件属性设置。

HWPF (可怕的字处理器格式) : 它是用来读取和写入MS-Word的文档扩展名的文件。

XWPF (XML字处理器格式) : 它是用来读取和写入MS-Word的docx扩展名的文件。

HSLF (可怕的幻灯片版式格式) : 它是用于读取，创建和编辑PowerPoint演示文稿。

HDGF (可怕的图表格式) : 它包含类和方法为MS-Visio的二进制文件。

HPBF (可怕的出版商格式) : 它被用来读取和写入MS-Publisher文件。

## 二、代码练习

**1 HSSF对xls后缀名的Excel进行读取内容**

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

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.poifs.filesystem.POIFSFileSystem;

import org.apache.poi.ss.usermodel.CellStyle;

import org.apache.poi.ss.usermodel.CellType;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

public class ReadExcelForHSSF {

public void read() {

File file = new File("././POI/POI1.xls");

if (!file.exists())

System.out.println("文件不存在");

try {

//1.读取Excel的对象

POIFSFileSystem poifsFileSystem = new POIFSFileSystem(new FileInputStream(file));

//2.Excel工作薄对象

HSSFWorkbook hssfWorkbook = new HSSFWorkbook(poifsFileSystem);

//3.Excel工作表对象

HSSFSheet hssfSheet = hssfWorkbook.getSheetAt(0);

//总行数

int rowLength = hssfSheet.getLastRowNum()+1;

//4.得到Excel工作表的行

HSSFRow hssfRow = hssfSheet.getRow(0);

//总列数

int colLength = hssfRow.getLastCellNum();

//得到Excel指定单元格中的内容

HSSFCell hssfCell = hssfRow.getCell(0);

//得到单元格样式

CellStyle cellStyle = hssfCell.getCellStyle();

for (int i = 0; i < rowLength; i++) {

//获取Excel工作表的行

HSSFRow hssfRow1 = hssfSheet.getRow(i);

for (int j = 0; j < colLength; j++) {

//获取指定单元格

HSSFCell hssfCell1 = hssfRow1.getCell(j);

//Excel数据Cell有不同的类型，当我们试图从一个数字类型的Cell读取出一个字符串时就有可能报异常：

//Cannot get a STRING value from a NUMERIC cell

//将所有的需要读的Cell表格设置为String格式

if (hssfCell1 != null) {

hssfCell1.setCellType(CellType.STRING);

}

//获取每一列中的值

System.out.print(hssfCell1.getStringCellValue() + "\t");

}

System.out.println();

}

} catch (IOException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

new ReadExcelForHSSF().read();

}

}

**2 XSSF对xlsx后缀名的Excel进行读取内容**

import org.apache.poi.ss.usermodel.\*;

import java.io.\*;

public class ReadExcelForXSSF {

public void read() {

File file = new File("././POI/POI2.xlsx");

InputStream inputStream = null;

Workbook workbook = null;

try {

inputStream = new FileInputStream(file);

workbook = WorkbookFactory.create(inputStream);

inputStream.close();

//工作表对象

Sheet sheet = workbook.getSheetAt(0);

//总行数

int rowLength = sheet.getLastRowNum()+1;

//工作表的列

Row row = sheet.getRow(0);

//总列数

int colLength = row.getLastCellNum();

//得到指定的单元格

Cell cell = row.getCell(0);

//得到单元格样式

CellStyle cellStyle = cell.getCellStyle();

System.out.println("行数：" + rowLength + ",列数：" + colLength);

for (int i = 0; i < rowLength; i++) {

row = sheet.getRow(i);

for (int j = 0; j < colLength; j++) {

cell = row.getCell(j);

//Excel数据Cell有不同的类型，当我们试图从一个数字类型的Cell读取出一个字符串时就有可能报异常：

//Cannot get a STRING value from a NUMERIC cell

//将所有的需要读的Cell表格设置为String格式

if (cell != null)

cell.setCellType(CellType.STRING);

//对Excel进行修改

if (i > 0 && j == 1)

cell.setCellValue("1000");

System.out.print(cell.getStringCellValue() + "\t");

}

System.out.println();

}

//将修改好的数据保存

OutputStream out = new FileOutputStream(file);

workbook.write(out);

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

new ReadExcelForXSSF().read();

}

}

**3 在这里对Excel写入内容使用的XSSF（HSSF也是同样的道理，不再多举例）：**

import org.apache.poi.hssf.util.HSSFColor;

import org.apache.poi.ss.usermodel.\*;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.\*;

public class WriteExcelForXSSF {

public void write() {

Workbook workbook = new XSSFWorkbook();

Sheet sheet = workbook.createSheet("0");

Row row = sheet.createRow(0);

CellStyle cellStyle = workbook.createCellStyle();

// 设置这些样式

cellStyle.setFillForegroundColor(HSSFColor.SKY\_BLUE.index);

cellStyle.setFillPattern(FillPatternType.SOLID\_FOREGROUND);

cellStyle.setBorderBottom(CellStyle.BORDER\_THIN);

cellStyle.setBorderLeft(CellStyle.BORDER\_THIN);

cellStyle.setBorderRight(CellStyle.BORDER\_THIN);

cellStyle.setBorderTop(CellStyle.BORDER\_THIN);

cellStyle.setAlignment(CellStyle.ALIGN\_CENTER);

row.createCell(0).setCellStyle(cellStyle);

row.createCell(0).setCellValue("姓名");

row.createCell(1).setCellStyle(cellStyle);

row.createCell(1).setCellValue("年龄");

workbook.setSheetName(0, "信息");

try {

File file = new File("././POI/POI3.xlsx");

FileOutputStream fileoutputStream = new FileOutputStream(file);

workbook.write(fileoutputStream);

fileoutputStream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

new WriteExcelForXSSF().write();

}

}