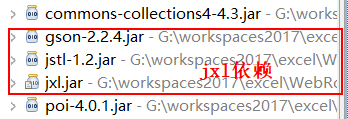
JXl 和 POI操作Excel

# 依赖



# 使用JXL读取excel内容

/\*\*

\* jxl 表格内容的读取

\* **@throws** BiffException

\* **@throws** IOException

\*/

**public** **static** **void** test1() **throws** BiffException, IOException{

File file = **new** File("G:\\you.xls");

InputStream is = **new** FileInputStream(file);

Workbook workBook = Workbook.*getWorkbook*(is);

**int** numberOfSheets = workBook.getNumberOfSheets();

**for**(**int** i = 0;i<numberOfSheets;i++){

Sheet sheet = workBook.getSheet(i);

String name = sheet.getName();

System.***out***.println(name);

**for**(**int** j =0;j<sheet.getRows();j++){

**for**(**int** h = 0;h<sheet.getColumns();h++){

String contents = sheet.getCell(h,j).getContents();

System.***out***.println(contents);

}

}

}

}

# 使用jxl读取并修改excel内容

/\*\*

\* jxl 表格内容 的读取和修改

\* **@throws** IOException

\* **@throws** BiffException

\* **@throws** RowsExceededException

\* **@throws** WriteException

\*/

**public** **static** **void** test2() **throws** IOException, BiffException, RowsExceededException, WriteException{

File file = **new** File("G:\\you.xls");

//获取模板文件

Workbook workBook = Workbook.*getWorkbook*(file);

//获取操作输出文件

WritableWorkbook writableWorkBook = Workbook.*createWorkbook*(**new** File("G:\\my.xls"),workBook);

//获取模板的文件页

Sheet sheet = workBook.getSheet(0);

//获取输出的文件页

WritableSheet sheet2 = writableWorkBook.getSheet(0);

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

**for**(**int** j = 0;j<sheet.getColumns();j++){

/\*\* 获取模板的 内容 和 格式 \*/

//获取单元格

Cell cell = sheet.getCell(j, i);

//获取单元格格式

CellFormat cellFormat = cell.getCellFormat();

//获取单元格内容

String contents = cell.getContents();

/\*\*单元格内容 和 格式 设置\*/

//创建内容标签 该label实现了WritableCell

jxl.write.Label label = **new** jxl.write.Label(j, i, "修改后的内容"+j+contents);

//设置单元格格式

label.setCellFormat(cellFormat);

//将标签添加到输出文件页中

sheet2.addCell(label);

// 自定义设置

/\*

WritableFont font = new WritableFont((jxl.format.Font) WritableFont.BOLD);

WritableCellFormat format = new WritableCellFormat(font);

label.setCellFormat(format);\*/

}

}

/\*\*内容写入与关闭\*/

//写入文件

writableWorkBook.write();

//关闭文件

writableWorkBook.close();

//关闭模板文件

workBook.close();

System.***out***.println("转换完毕");

}

# 使用poi读取并修改excel内容

/\*\*

\* poi 内容读取和修改

\* **@throws** IOException

\*/

**public** **static** **void** test3() **throws** IOException{

// 文件输入流

InputStream is = **new** FileInputStream(**new** File("G:\\you.xls"));

// 创建输出流

OutputStream os = **new** FileOutputStream(**new** File("G:\\my.xls"));

// 获取工作薄

HSSFWorkbook workBook = **new** HSSFWorkbook(is);

// 获取内容页

HSSFSheet sheet = workBook.getSheetAt(0);

// 获取内容页名称

String sheetName = sheet.getSheetName();

System.***out***.println(sheetName);

//获取最后一行的下标

**int** lastRowNum = sheet.getLastRowNum();

**for**(**int** i = 0;i<=lastRowNum;i++){

// 获取行对象

HSSFRow row = sheet.getRow(i);

// 获取该行的最后一列下标

**short** lastCellNum = row.getLastCellNum();

// 遍历该该行的单元格

**for**(**int** j = 0;j<lastCellNum;j++){

// 获取单元格对象

HSSFCell cell = row.getCell(j);

// 设置单元格的类型 为字符串： 以能够读取字符串值

cell.setCellType(CellType.***STRING***);

// 获取内容呢

String stringCellValue = cell.getStringCellValue();

System.***out***.println(stringCellValue);

// 修改内容

cell.setCellValue("修改后的内荣："+j+" : "+stringCellValue);

// 设置格式

HSSFCellStyle cellStyle = workBook.createCellStyle();

//cellStyle.setBorderBottom(border);

}

}

// 输出修改的内容呢

workBook.write(os);

workBook.close();

System.***out***.println("输出完毕");

}

## 可能异常

### 异常1

/\*

\* POI操作Excel时偶尔会出现Cannot get a text value from a numeric cell的异常错误。

异常原因：Excel数据Cell有不同的类型，当我们试图从一个数字类型的Cell读取出一个字符串并写入数据库时，就会出现Cannot get a text value from a numeric cell的异常错误。

此异常常见于类似如下代码中：row.getCell(0).getStringCellValue()；

解决办法：先设置Cell的类型，然后就可以把纯数字作为String类型读进来了：

if(row.getCell(0)!=null){

          row.getCell(0).setCellType(CellType.STRING);

          stuUser.setPhone(row.getCell(0).getStringCellValue());

   }

\* \*/

### 异常2

类无法找到 ：org.apache.commons.collections4.bidimap.TreeBidiMap.class

解决，添加依赖jar包 org.apache.commons

# 单元格设置

## jxl操作

### 样式设置

//1 列的宽度设置

sheet1.setColumnView(0, 30); // 设置列的宽度

sheet1.setColumnView(1, 30); // 设置列的宽度

sheet1.setColumnView(2, 30); // 设置列的宽度

sheet1.setColumnView(3, 30); // 设置列的宽度

sheet1.setColumnView(4, 30); // 设置列的宽度

sheet1.setColumnView(5, 30); // 设置列的宽度

//2 行的高度设置

sheet1.setRowView(0, 1000); // 设置行的高度

sheet1.setRowView(1, 500); // 设置行的高度

/\*\*

\*3 定义单元格样式 : 字体 格式 粗体 斜体 下划线 颜色

\*/

WritableFont wf\_title = new WritableFont(WritableFont.ARIAL, 20,

WritableFont.BOLD, false, UnderlineStyle.NO\_UNDERLINE,

jxl.format.Colour.RED); // 定义格式 字体 下划线 斜体 粗体 颜色

WritableFont wf\_head = new WritableFont(WritableFont.ARIAL, 10,

WritableFont.BOLD, false, UnderlineStyle.NO\_UNDERLINE,

jxl.format.Colour.GREEN); // 定义格式 字体 下划线 斜体 粗体 颜色

WritableFont wf\_table = new WritableFont(WritableFont.ARIAL, 8,

WritableFont.BOLD, false, UnderlineStyle.NO\_UNDERLINE,

jxl.format.Colour.BLACK); // 定义格式 字体 下划线 斜体 粗体 颜色

WritableCellFormat wcf\_title = new WritableCellFormat(wf\_title); // 单元格定义

wcf\_title.setBackground(jxl.format.Colour.BLACK); // 设置单元格的背景颜色

wcf\_title.setAlignment(jxl.format.Alignment.CENTRE); // 设置对齐方式

WritableCellFormat wcf\_head = new WritableCellFormat(wf\_head);

wcf\_head.setBackground(jxl.format.Colour.BLACK);

wcf\_head.setAlignment(jxl.format.Alignment.CENTRE);

WritableCellFormat wcf\_table = new WritableCellFormat(wf\_table);

wcf\_table.setBackground(jxl.format.Colour.BLACK);

wcf\_table.setAlignment(jxl.format.Alignment.CENTRE);

/\*\*

\* 使用样式的单元格

\*/

// 1.添加Label对象三个参数意思：【列，行，值】

sheet1.addCell(new Label(0, 0, "标题", wcf\_title)); // 普通的带有定义格式的单元格

sheet1.addCell(new Label(0, 1, "表头1", wcf\_head));

sheet1.addCell(new Label(1, 1, "表头2", wcf\_head));

sheet1.addCell(new Label(2, 1, "表头3", wcf\_head));

sheet1.addCell(new Label(3, 1, "表头4", wcf\_head));

sheet1.addCell(new Label(4, 1, "表头5", wcf\_head));

sheet1.addCell(new Label(5, 1, "表头6", wcf\_head));

### 单元格合并 以及格式设置：字体 背景 列宽 行高 边框 对齐 自动换行

//.mergeCells(int col1, int row1, int col2, int row2)

sheet1.mergeCells(0, 0, 5, 0); // 合并单元格

/\*\*

\* jxl 合并单元格

\* **@throws** BiffException

\* **@throws** IOException

\* **@throws** RowsExceededException

\* **@throws** WriteException

\*/

**public** **static** **void** test4() **throws** BiffException, IOException, RowsExceededException, WriteException{

//创建可编辑的工作薄

WritableWorkbook writeBook = Workbook.*createWorkbook*(**new** File("G:\\my.xls"));

//创建第一页

WritableSheet sheet = writeBook.createSheet("第一页", 0);

//工作页格式设置

/\* 设置方式一

CellView view = new CellView();

view.setAutosize(true);

view.setSize(100);

sheet.setColumnView(0, view );

\*/

sheet.setRowView(0, 50);

sheet.setColumnView(0, 50);

//创建内容标签1

jxl.write.Label label = **new** jxl.write.Label(0, 0 ,"手机品牌1");

//格式化单元格

WritableCellFormat wcf = **new** WritableCellFormat();

//1 背景设置

wcf.setBackground(Colour.***RED***);

//2 自动换行

wcf.setWrap(**true**);

//3 水平对齐方式

wcf.setAlignment(Alignment.*CENTRE*);

//4 垂直对齐方式

wcf.setVerticalAlignment(VerticalAlignment.*CENTRE*);

//5 字体设置

WritableFont font = **new** WritableFont(WritableFont.***ARIAL***, 20, WritableFont.***BOLD***, **true**, UnderlineStyle.***DOUBLE\_ACCOUNTING***, Colour.***ORANGE***);

wcf.setFont(font );

//6 设置边框

wcf.setBorder(Border.***BOTTOM***, BorderLineStyle.***DOUBLE***, Colour.***BLUE\_GREY***);

//单元格使用样式

label.setCellFormat(wcf);

//添加标签到工作薄

sheet.addCell(label);

//合并单元格

sheet.mergeCells(0, 0, 0, 3);

//创建内容标签2

jxl.write.Label label2 = **new** jxl.write.Label(1, 0, "苹果");

jxl.write.Label label3 = **new** jxl.write.Label(1, 1, "三星");

jxl.write.Label label4 = **new** jxl.write.Label(1, 2, "华为");

jxl.write.Label label5 = **new** jxl.write.Label(1, 3, "VIVO");

sheet.addCell(label2);

sheet.addCell(label3);

sheet.addCell(label4);

sheet.addCell(label5);

writeBook.write();

writeBook.close();

}



### 综合

public static String testEx() {

// a, 计算路径

SimpleDateFormat sdf = new SimpleDateFormat("/yyyy-MM-dd/");

String datePath = sdf.format(new Date());

datePath = "中国石油西部管道输气量交接电子文档" + datePath;

String uuidFileName = UUID.randomUUID().toString();

// 创建文件夹

File dir = new File(Globals.fielImportPath + datePath);

if (!dir.exists()) {

dir.mkdirs();

}

String path = Globals.fielImportPath + datePath + uuidFileName + "."

+ "xls";

try {

WritableWorkbook wwb = Workbook.createWorkbook(new File(

path));

WritableSheet sheet1= wwb.createSheet("前五个站队", 0);

WritableSheet sheet2 = wwb.createSheet("后两个站队", 1);

WritableSheet sheet3 = wwb.createSheet("后三个站队", 2);

sheet1.setColumnView(0, 30); // 设置列的宽度

sheet1.setColumnView(1, 30); // 设置列的宽度

sheet1.setColumnView(2, 30); // 设置列的宽度

sheet1.setColumnView(3, 30); // 设置列的宽度

sheet1.setColumnView(4, 30); // 设置列的宽度

sheet1.setColumnView(5, 30); // 设置列的宽度

sheet1.setRowView(0, 1000); // 设置行的高度

sheet1.setRowView(1, 500); // 设置行的高度

/\*\*

\* 定义单元格样式

\*/

WritableFont wf\_title = new WritableFont(WritableFont.ARIAL, 20,

WritableFont.BOLD, false, UnderlineStyle.NO\_UNDERLINE,

jxl.format.Colour.RED); // 定义格式 字体 下划线 斜体 粗体 颜色

WritableFont wf\_head = new WritableFont(WritableFont.ARIAL, 10,

WritableFont.BOLD, false, UnderlineStyle.NO\_UNDERLINE,

jxl.format.Colour.GREEN); // 定义格式 字体 下划线 斜体 粗体 颜色

WritableFont wf\_table = new WritableFont(WritableFont.ARIAL, 8,

WritableFont.BOLD, false, UnderlineStyle.NO\_UNDERLINE,

jxl.format.Colour.BLACK); // 定义格式 字体 下划线 斜体 粗体 颜色

WritableCellFormat wcf\_title = new WritableCellFormat(wf\_title); // 单元格定义

wcf\_title.setBackground(jxl.format.Colour.BLACK); // 设置单元格的背景颜色

wcf\_title.setAlignment(jxl.format.Alignment.CENTRE); // 设置对齐方式

WritableCellFormat wcf\_head = new WritableCellFormat(wf\_head);

wcf\_head.setBackground(jxl.format.Colour.BLACK);

wcf\_head.setAlignment(jxl.format.Alignment.CENTRE);

WritableCellFormat wcf\_table = new WritableCellFormat(wf\_table);

wcf\_table.setBackground(jxl.format.Colour.BLACK);

wcf\_table.setAlignment(jxl.format.Alignment.CENTRE);

/\*\*

\* 使用样式的单元格

\*/

// 1.添加Label对象三个参数意思：【列，行，值】

sheet1.addCell(new Label(0, 0, "标题", wcf\_title)); // 普通的带有定义格式的单元格

sheet1.addCell(new Label(0, 1, "表头1", wcf\_head));

sheet1.addCell(new Label(1, 1, "表头2", wcf\_head));

sheet1.addCell(new Label(2, 1, "表头3", wcf\_head));

sheet1.addCell(new Label(3, 1, "表头4", wcf\_head));

sheet1.addCell(new Label(4, 1, "表头5", wcf\_head));

sheet1.addCell(new Label(5, 1, "表头6", wcf\_head));

//.mergeCells(int col1, int row1, int col2, int row2)

sheet1.mergeCells(0, 0, 5, 0); // 合并单元格

// 写入Exel工作表

wwb.write();

// 关闭Excel工作薄对象

wwb.close();

} catch (Exception e) {

e.printStackTrace();

return "0";

}

return path;

}

public static void main(String[] args) {

UtilIO uio=new UtilIO();

System.out.println(uio.testEx());

}

## poi操作

### poi 合并 单元格

XSSFWorkbook wb = new XSSFWorkbook();

XSSFSheet sheet = wb.createSheet();

//这个就是合并单元格

//参数说明：1：开始行 2：结束行 3：开始列 4：结束列

//比如我要合并 第二行到第四行的 第六列到第八列 sheet.addMergedRegion(new CellRangeAddress(1,3,5,7));

sheet.addMergedRegion(new CellRangeAddress(0,0,0,1));

XSSFRow row = sheet.createRow(number);

需要注意的地方（不对的地方请指教）

感觉唯一要注意的地方就是：需要先设置 合并单元格，然后再 生成 行。

比如我们要生成的单元格为：



代码可以这样写：

sheet.addMergedRegion(new CellRangeAddress(0,3,0,0));

sheet.addMergedRegion(new CellRangeAddress(0,3,3,3));

sheet.addMergedRegion(new CellRangeAddress(0,3,4,4));

//第一行数据

XSSFRow row = sheet.createRow(0);

row.createCell(0).setCellValue("工作站");

row.createCell(1).setCellValue("位置");

row.createCell(2).setCellValue("序号");

row.createCell(3).setCellValue("订单号");

row.createCell(4).setCellValue("成品号/型号");

//第二行数据

XSSFRow row = sheet.createRow(number);

//row.createCell(0).setCellValue("工作站");//因为和上面的行合并了，所以不用再次 赋值了

row.createCell(1).setCellValue("位置");

row.createCell(2).setCellValue("序号");

//row.createCell(3).setCellValue("订单号");//因为和上面的行合并了，所以不用再次 赋值了

//row.createCell(4).setCellValue("成品号/型号");//因为和上面的行合并了，所以不用再次 赋值了



### 综合样式设置

使用POI创建一个简单的   myXls.xls   文件     
      
常用的包为   org.apache.poi.hssf.usermodel.\*;     
例子:         
import   java.io.\*;     
import   org.apache.poi.hssf.usermodel.\*;         
public   class   ZoomSheet   {        
          public   ZoomSheet()   {     
          }      
          public   static   void   main(String   args[])  throws   IOException   {     
                  HSSFWorkbook   wb   =   new   HSSFWorkbook();     
                  HSSFSheet   sheet1   =   wb.createSheet("new   sheet");     
                  FileOutputStream   fileOut   =   new   FileOutputStream("workbook.xls");     
                  wb.write(fileOut);     
                  fileOut.close();     
          }     
}   

常用类:     
1 HSSFWorkbook

创建   xls   的对象; HSSFWorkbook   hw   =   new   HSSFWorkbook();                     
设置分区显示; hw.setRepeatingRowsAndColumns(sheet的index,   行,   列,   行,   列);   

2 HSSFSheet

创建   xls   中的sheet(工作表); HSSFSheet   sheet   =   hw.createSheet("sheet1");

sheet1   是   sheet   的名称可缺省     
设置列宽; sheet.setColumnWidth((short)short,   (short)short);   

3 HSSFRow

创建   xls   中的行; HSSFRow   row   =   sheet.createRow(0);            0   表示第一行     
设置行高; row.setHeight((short)short);   

4 HSSFFont

创建   xls   中的字体; HSSFFont   font   =   hw.createFont();     
设定字体大小; font.setFontHeightInPoints((short)54);     
设定为斜体; font.setItalic(true);     
设定文字删除线; font.setStrikeout(true);   

5 HSSFCellStyle

设定单元格风格; HSSFCellStyle   style   =   wb.createCellStyle();     
加入字体; style.setFont(font);     
单元格水平对齐方式; style.setAlignment(align);   //单元格水平   0普通   1左对齐   2 居中   3 右对齐   4 填充   5 正当   6 居中选择     
单元格垂直对齐方式; style.setVerticalAlignment(align);       //单元格垂直   0   居上   1   居中   2   居下   3   正当     
单元格下边框为细线; style.setBorderBottom((short)short);     
同上一命令一同使用,设置颜色; style.setBottomBorderColor((short)short);     
单元格左边框; style.setBorderLeft((short)short);     
 style.setLeftBorderColor((short)short);     
单元格右边框; style.setBorderRight((short)short);     
 style.setRightBorderColor((short)short);     
单元格上边框; style.setBorderTop((short)short);     
 style.setTopBorderColor((short)short);

6 HSSFCell

设定单元格; HSSFCell   cell   =   row.createCell((short)0);

单元格字符编号(中文); cell.setEncoding(HSSFCell.ENCODING\_UTF\_16);       //中文     
单元格显示的值; cell.setCellValue("中医药");     值的类型有:double,int,String,Date,boolean     
单元格背景色; style.setFillForegroundColor((short)short);     
图案类型; style.setFillPattern((short)short);     
单元格合并; sheet.addMergedRegion(new   Region(行,   (short)列,   行,   (short)列));     
      
单元格风格加入; cell.setCellStyle(style);     
      
                  
      
      
7 打印设置     
引入包 import   org.apache.poi.hssf.usermodel.HSSFPrintSetup;     
创建打印设置对象 HSSFPrintSetup   hps   =   hs.getPrintSetup();     
设置A4纸 hps.setPaperSize((short)9);         
将页面设置为横向打印模式 hps.setLandscape(true);       
      
设置打印页面为水平居中 sheet.setHorizontallyCenter(true);           
设置打印页面为垂直居中 sheet.setVerticallyCenter(true);

8 单元格内换行设置

网上找到的文章都是说在excel里的文字里加上\n,\n\r,\r\n之类，反正各种各样的都有，更奇怪的是还有人说在单元格里加上<br>   
后来我试过用\r后的效里是生成的文件里，你用打开时，并不会换行，如果你用鼠标在单元格里点一下之后就会自动换行。

可以通过如下方式进行，  
1. 首先在需要强制换行的单元格里使用poi的样式，并且把样式设定为自动换行   
   # HSSFCellStyle cellStyle=workbook.createCellStyle();       
   # cellStyle.setWrapText(true);       
   # cell.setCellStyle(cellStyle);   
2. 其次是在需要强制换行的单元格，使用\r\n就可以实再强制换行   
   1. HSSFCell cell = row.createCell((short)0);   
   2. cell.setCellStyle(cellStyle);

3. cell.setCellValue(new HSSFRichTextString("hello\r\n world!"));   
这样就能实现强制换行，  
换行后的效里是单元格里强制换行  
hello  
world!

### [POI做导出Excel设置单元格中字体大小颜色，合并行列](http://blog.csdn.net/kunkun378263/article/details/9040639)

public class ExcelTest {  
  /\*\*     
  \* @param args     
  \*/     
  public static void main(String[] args) throws IOException {      
   
      try {      
          HSSFWorkbook wb = new HSSFWorkbook();      
          HSSFSheet sheet = wb.createSheet("new   sheet");      
          HSSFCellStyle style = wb.createCellStyle(); // 样式对象      
   
          style.setVerticalAlignment(HSSFCellStyle.VERTICAL\_CENTER);// 垂直      
          style.setAlignment(HSSFCellStyle.ALIGN\_CENTER);// 水平      
          /\*\*字体begin\*/  
          style.setFillForegroundColor(HSSFColor.SKY\_BLUE.index);   
          //背景颜色  
//          style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND);  
//          style.setBorderBottom(HSSFCellStyle.BORDER\_THIN);  
//          style.setBorderLeft(HSSFCellStyle.BORDER\_THIN);  
//          style.setBorderRight(HSSFCellStyle.BORDER\_THIN);  
//          style.setBorderTop(HSSFCellStyle.BORDER\_THIN);  
//          style.setAlignment(HSSFCellStyle.ALIGN\_CENTER);  
            
          //生成一个字体  
          HSSFFont font=wb.createFont();  
          font.setColor(HSSFColor.BLACK.index);//HSSFColor.VIOLET.index //字体颜色  
          font.setFontHeightInPoints((short)12);  
          font.setBoldweight(HSSFFont.BOLDWEIGHT\_BOLD);         //字体增粗  
         //把字体应用到当前的样式  
          style.setFont(font);  
          /\*\*字体end\*/  
          HSSFRow row = sheet.createRow((short) 0);      
          HSSFRow row2 = sheet.createRow((short) 1);      
            
          // 四个参数分别是：起始行，起始列，结束行，结束列      
          sheet.addMergedRegion(new Region(0, (short) 0, 5, (short) 0));      
          HSSFCell ce = row.createCell((short) 0);      
          ce.setCellValue("项目\\日期"); // 表格的第一行第一列显示的数据      
          ce.setCellStyle(style); // 样式，居中      
          int num = 0;      
          for (int i = 0; i < 9; i++) { // 循环9次，每一次都要跨单元格显示      
              // 计算从那个单元格跨到那一格      
              int celln = 0;      
              int celle = 0;      
              if (i == 0) {      
                  celln = 0;      
                  celle = 1;      
              } else {      
                  celln = (i \* 2);      
                  celle = (i \* 2 + 1);      
              }      
              // 单元格合并      
              // 四个参数分别是：起始行，起始列，结束行，结束列      
              sheet.addMergedRegion(new Region(0, (short) (celln + 1), 0,      
                      (short) (celle + 1)));      
              HSSFCell cell = row.createCell((short) (celln + 1));      
              cell.setCellValue("merging" + i); // 跨单元格显示的数据      
              cell.setCellStyle(style); // 样式      
              // 不跨单元格显示的数据，如：分两行，上一行分别两格为一格，下一行就为两格，“数量”，“金额”      
              HSSFCell cell1 = row2.createCell((short) celle);      
              HSSFCell cell2 = row2.createCell((short) (celle + 1));      
              cell1.setCellValue("数量");      
              cell1.setCellStyle(style);      
              cell2.setCellValue("金额");      
              cell2.setCellStyle(style);      
              num++;      
          }      
   
          // 在后面加上合计百分比      
   
          // 合计 在最后加上，还要跨一个单元格     //四个参数分别是：起始行，起始列，结束行，结束列      
          sheet.addMergedRegion(new Region(0, (short) (2 \* num + 1), 0,      
                  (short) (2 \* num + 2)));      
          HSSFCell cell = row.createCell((short) (2 \* num + 1));      
          cell.setCellValue("合计");      
          cell.setCellStyle(style);    
          HSSFCell cell1 = row2.createCell((short) (2 \* num + 1));      
          HSSFCell cell2 = row2.createCell((short) (2 \* num + 2));      
          cell1.setCellValue("数量");      
          cell1.setCellStyle(style);      
          cell2.setCellValue("金额");      
          cell2.setCellStyle(style);      
   
          // 百分比 同上      
          sheet.addMergedRegion(new Region(0, (short) (2 \* num + 3), 0,      
                  (short) (2 \* num + 4)));      
          HSSFCell cellb = row.createCell((short) (2 \* num + 3));      
          cellb.setCellValue("百分比");      
          cellb.setCellStyle(style);      
          HSSFCell cellb1 = row2.createCell((short) (2 \* num + 3));      
          HSSFCell cellb2 = row2.createCell((short) (2 \* num + 4));      
          cellb1.setCellValue("数量");      
          cellb1.setCellStyle(style);      
          cellb2.setCellValue("金额");      
          cellb2.setCellStyle(style);    
            
          //输出一些数据 然后再输出表头  
            
          FileOutputStream fileOut = new FileOutputStream("D://workbook.xls");     
          wb.write(fileOut);      
          fileOut.close();      
          System.out.print("OK");      
      } catch (Exception ex) {      
          ex.printStackTrace();      
      }      
   
  }    

### //设置单元格字体颜色

import [Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "_blank).io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

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

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

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

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

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

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

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

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

public class test {

public static void main(String[] args) {

Workbook workbook = new HSSFWorkbook();

Sheet sheet = workbook.createSheet();

Cell cell = sheet.createRow(0).createCell(0);

CreationHelper helper = workbook.getCreationHelper();

RichTextString str = helper.createRichTextString("a\nb\nc\nd\ne\n");// 在这里使用\n表示回车

Font[] fonts = new Font[5];

fonts[0] = workbook.createFont();

fonts[0].setColor(HSSFColor.YELLOW.index);

fonts[1] = workbook.createFont();

fonts[1].setColor(HSSFColor.RED.index);

fonts[2] = workbook.createFont();

fonts[2].setColor(HSSFColor.BLUE.index);

f

onts[3] = workbook.createFont();

fonts[3].setColor(HSSFColor.ROSE.index);

fonts[4] = workbook.createFont();

fonts[4].setColor(HSSFColor.BLACK.index);

for (int i = 0; i < 5; i++) {

str.applyFont(i \* 2, (i + 1) \* 2, fonts[i]);

}

cell.setCellValue(str);

try {

FileOutputStream out = new FileOutputStream(new File("d:\\1.xls"));

workbook.write(out);

out.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}