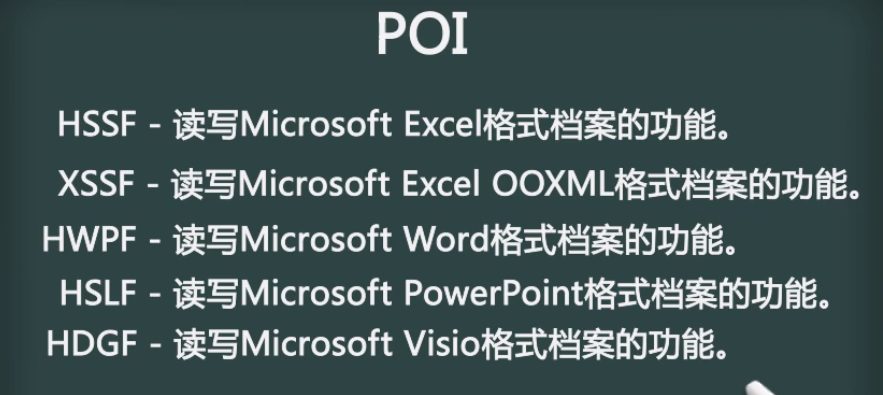
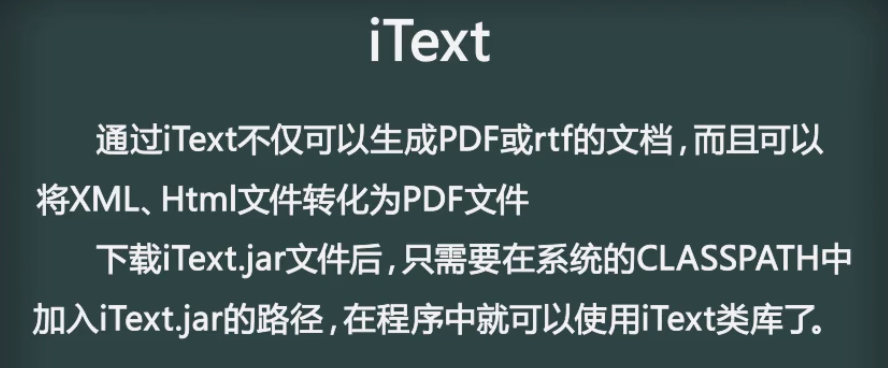
# 读写Excel文件常用的技术

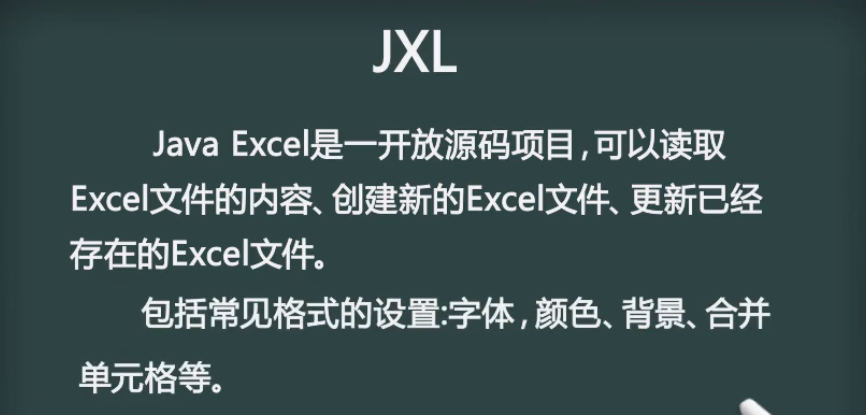
POI

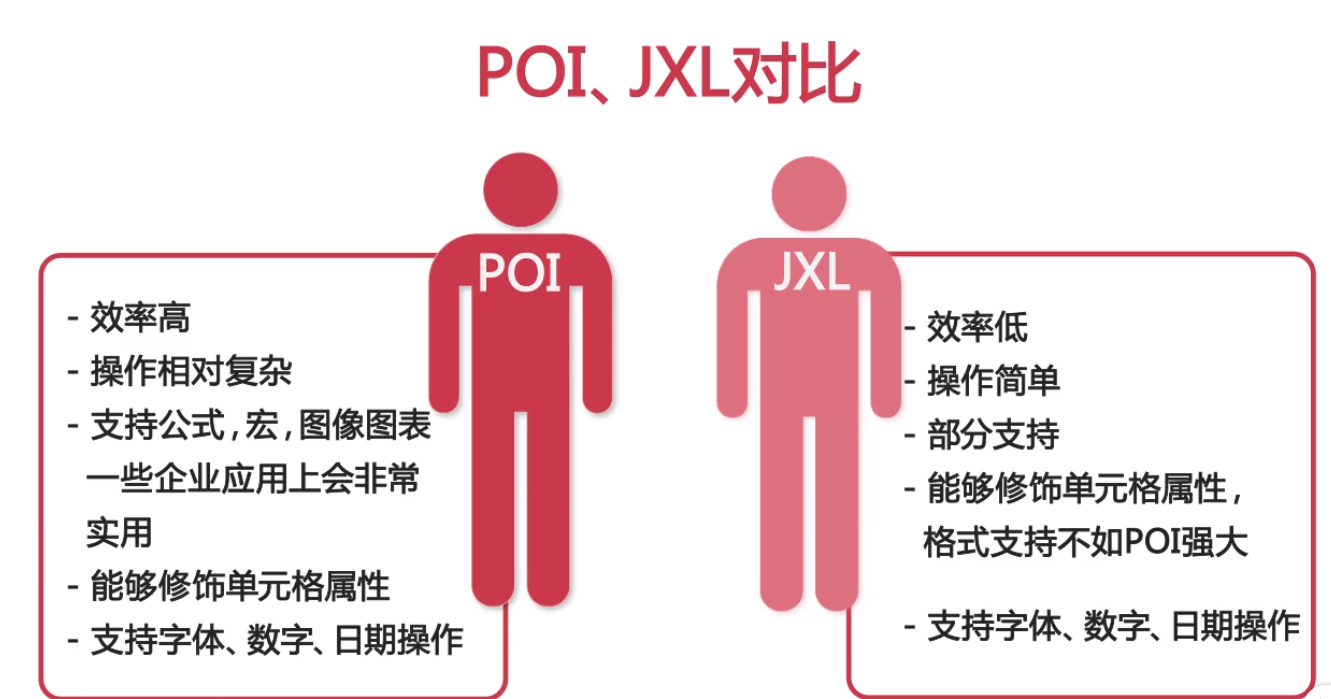


读写pdf文件



JXL

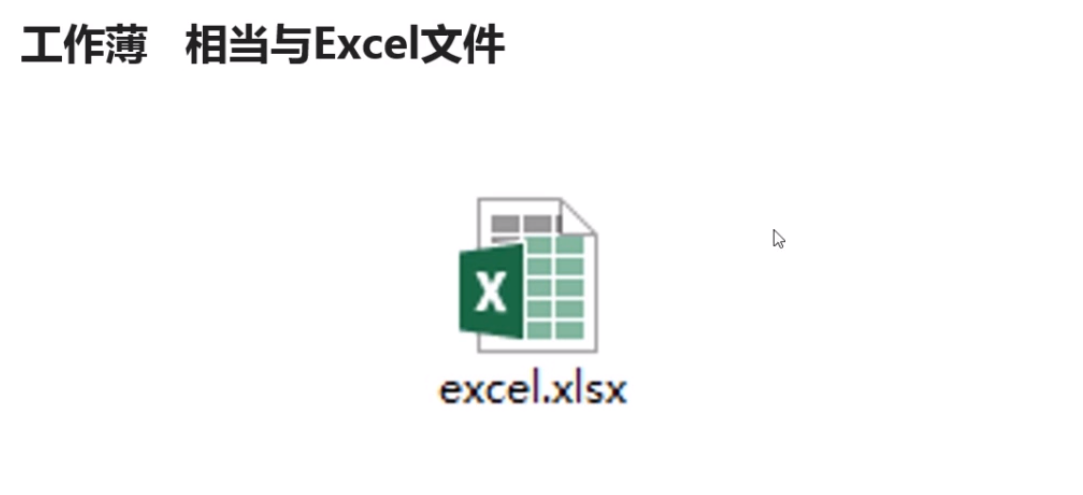


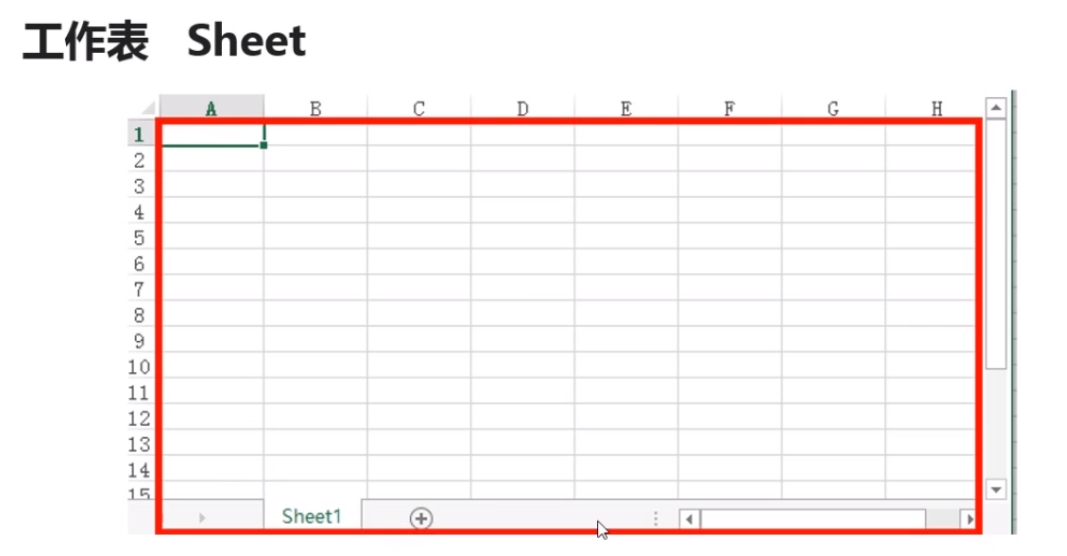


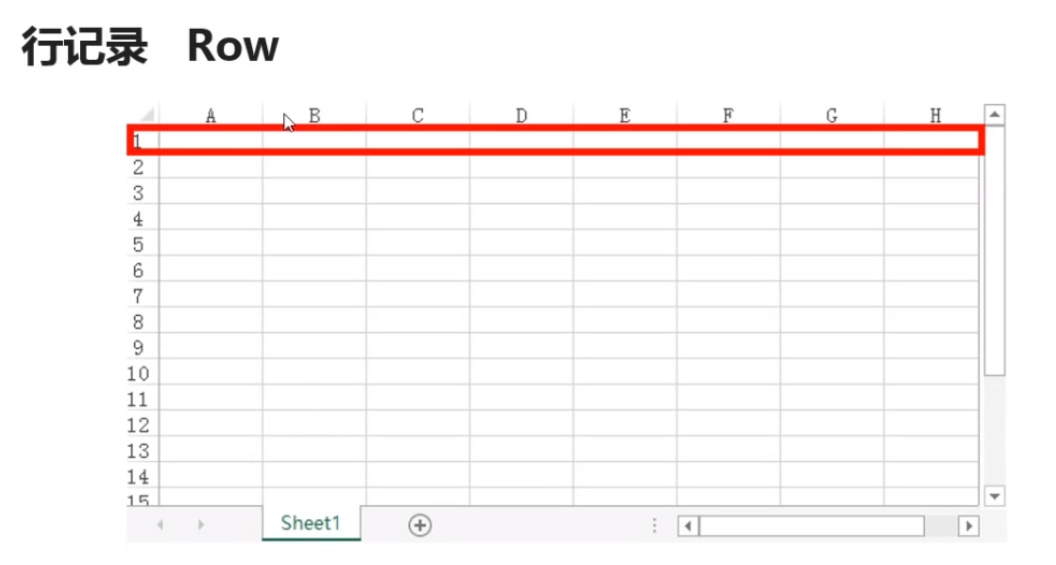
FASTEXCEL（一般不用）

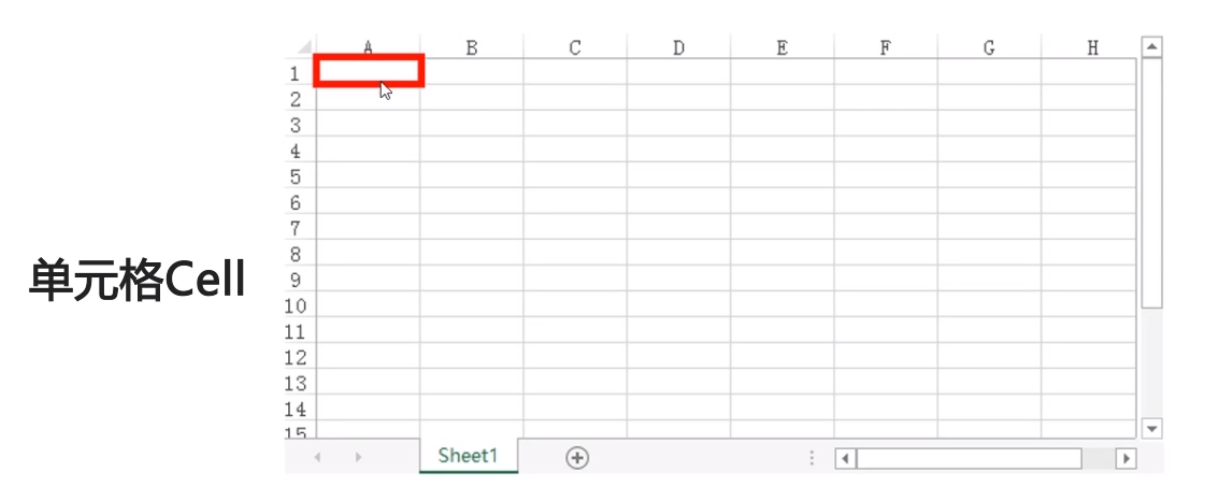


# excel基础知识









# POI解析EXCEL文件

代码示例：



# 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导出大量数据的处理方法

参考的网址：https://blog.csdn.net/u010003835/article/details/51590101

## 方法一：使用高版本的POI，具有内存自动释放的特点

POI之前的版本不支持[大数据](http://lib.csdn.net/base/20)量处理，如果数据过多则经常报OOM错误，有时候调整JVM大小效果也不是太好。3.8版本的POI新出来了SXSSFWorkbook,可以支持大数据量的操作，只是SXSSFWorkbook只支持.xlsx格式，不支持.xls格式。

3.8版本的POI对excel的导出操作，一般只使用HSSFWorkbook以及SXSSFWorkbook，HSSFWorkbook用来处理较少的数据量，SXSSFWorkbook用来处理大数据量以及超大数据量的导出。

HSSFWorkbook的使用方法和之前的版本的使用方法一致，这里就不在陈述使用方法了。

Workbook wb = new SXSSFWorkbook(100);

Sheet sh = wb.createSheet();

for(int rownum = 0; rownum < 100000; rownum++){

Row row = sh.createRow(rownum);

for(int cellnum = 0; cellnum < 10; cellnum++){

Cell cell = row.createCell(cellnum);

String address = new CellReference(cell).formatAsString();

cell.setCellValue(address);

}

}

FileOutputStream out = new FileOutputStream("/temp/sxssf.xlsx");

wb.write(out);

System.out.println("导出成功");

out.close();

## 方法二：将数据导出到多个Excel中，并进行压缩处理，上传到服务器中

我们知道在POI导出Excel时，数据量大了，很容易导致内存溢出，由于Excel 一个sheet允许的最大行数是65536这时我们想到分sheet进行导出；但是这种情况也不能解决内存溢出的问题。毕竟数据还是一次性在内存中进行保存的。首先：我们要确定数据量有多大，然后确定一个excel导出多少条数据，这样就可以确定导出的Excel的数量，于是我们就可以循环的导出excel并保存在任意的临时目录中。如果内存不够的话虚拟机就会去进行回收已经保存的excel在内存中的空间。假设我们已经成功的生成了多个excel，这时我们怎么把这N个excel文档传到客户端呢？其实一个一个的传也未尝不可，但是考虑那样对用户来说体验不够好，再次多个文件在网络上传输也比较慢。我们可以考虑对生成的几个文件进行压缩，然后传到客户端。

总结一下：1、分批次生成excel文件。2、压缩后到客户端。

示例：

# 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

## 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

## 创建行

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

## 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

## 合并单元格

sheet.addMergedRegion(new CellRangeAddress(int firstRow, int lastRow, int firstCol, int lastCol)

## 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());//给单元格设置日期

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

或者

cell.setCellValue(Calendar.getInstance());

cell.setCellStyle(cellstyle);

## 设置单元格的值

row.createCell(0).setCellValue();

## 获取行的总数

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

HSSFWorkbook hssf = new HSSFWorkbook(fs);

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

int rownum = sheet.getLastRowNum();

## 获取列的总数

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

HSSFWorkbook hssf = new HSSFWorkbook(fs);

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

HSSFRow hs = sheet.getRow(0);//得到行

int columnnum = hs.getLastCellNum();

## 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();

## 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

//设置单元格宽度

sheet.autoSizeColumn(2);

sheet.setDefaultColumnWidth((short) 20);

# 遇到的问题

## Package should contain a content type part [M1.13]

解决方法：这个错误的原因是读取的excel文件是加密文件，需要把密码取消了就可以了。