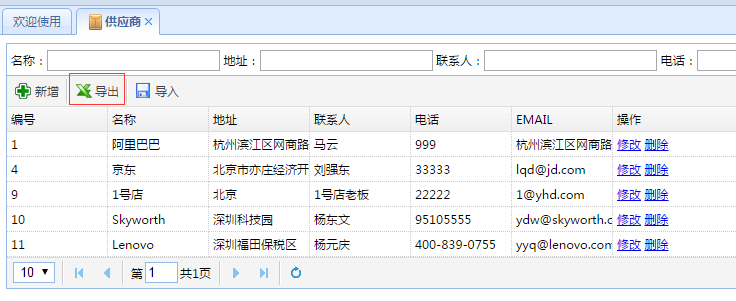
# ERP项目笔记 Day-10

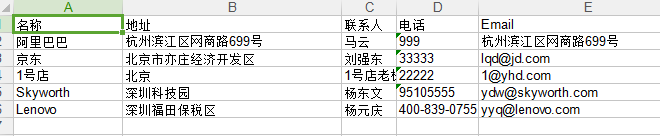
## 数据的导入导出

1. 供应商或客户数据导出
   1. 需求分析

点击导出按钮，将供应商或客户的信息导出为excel文档



下载到本地，打开导出的excel:



* 1. POI简介

[Apache](http://baike.baidu.com/item/Apache/6265" \t "http://baike.baidu.com/item/_blank) POI 是用[Java](http://baike.baidu.com/item/Java/85979" \t "http://baike.baidu.com/item/_blank)编写的免费开源的跨平台的 Java API，Apache POI提供API给Java程式对[Microsoft Office](http://baike.baidu.com/item/Microsoft Office" \t "http://baike.baidu.com/item/_blank)格式档案读和写的功能。POI为“Poor Obfuscation Implementation”的首字母缩写，意为“可怜的模糊实现”。

用它可以使用Java读取和创建,修改MS [Excel](http://baike.baidu.com/item/Excel/9564" \t "http://baike.baidu.com/item/_blank)文件.而且,还可以使用Java读取和创建MS [Word](http://baike.baidu.com/item/Word/2970534" \t "http://baike.baidu.com/item/_blank)和MS[PowerPoint](http://baike.baidu.com/item/PowerPoint" \t "http://baike.baidu.com/item/_blank)文件。Apache POI 提供Java操作Excel解决方案（适用于Excel97-2008）

**结构：**

**HSSF****- 提供读写[Microsoft Excel](http://baike.baidu.com/item/Microsoft Excel" \t "http://baike.baidu.com/item/_blank) [XLS](http://baike.baidu.com/item/XLS" \t "http://baike.baidu.com/item/_blank)格式档案的功能。**

XSSF  - 提供读写Microsoft Excel OOXML [XLSX](http://baike.baidu.com/item/XLSX" \t "http://baike.baidu.com/item/_blank)格式档案的功能。

HWPF  - 提供读写[Microsoft Word](http://baike.baidu.com/item/Microsoft Word" \t "http://baike.baidu.com/item/_blank) [DOC](http://baike.baidu.com/item/DOC" \t "http://baike.baidu.com/item/_blank)格式档案的功能。

HSLF  - 提供读写Microsoft [PowerPoint](http://baike.baidu.com/item/PowerPoint" \t "http://baike.baidu.com/item/_blank)格式档案的功能。

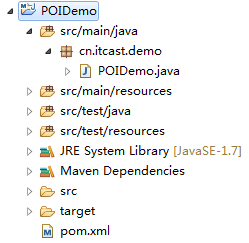
HDGF  - 提供读[Microsoft Visio](http://baike.baidu.com/item/Microsoft Visio" \t "http://baike.baidu.com/item/_blank)格式档案的功能。

HPBF  - 提供读[Microsoft Publisher](http://baike.baidu.com/item/Microsoft Publisher" \t "http://baike.baidu.com/item/_blank)格式档案的功能。

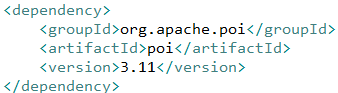
HSMF  - 提供读[Microsoft Outlook](http://baike.baidu.com/item/Microsoft Outlook" \t "http://baike.baidu.com/item/_blank)格式档案的功能。

* 1. 入门Demo
     1. 创建工程引入POI

1. 创建POIDemo工程

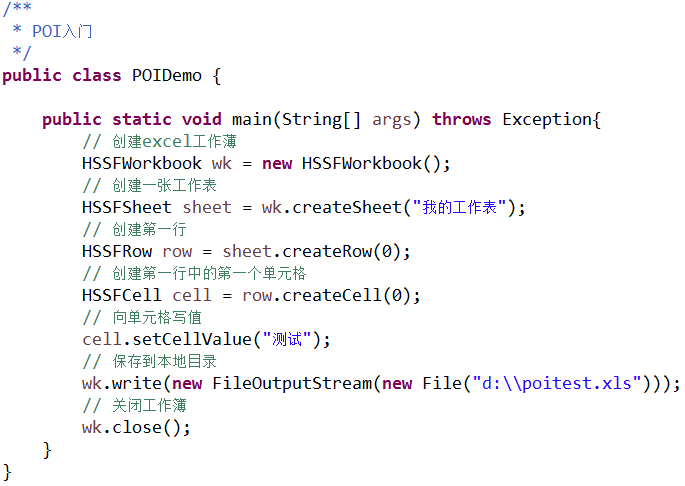


1. 引入POI依赖



* + 1. 向单元格写值

创建POIDemo类，代码如下：



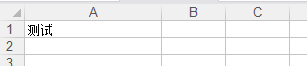
注：行和列的索引都是从0开始的，即0行表示实际上的第1行，0列表示第一列（A列）

* + 1. 设置列宽度

在上例中，加入：

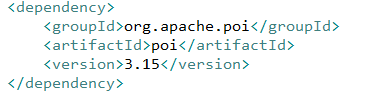


结果：



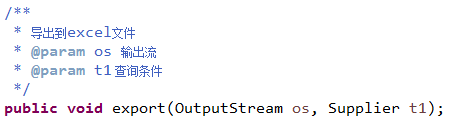
* 1. 后端代码编写
     1. 添加POI依赖

在父工程的pom.xml中添加POI的依赖



* + 1. 业务逻辑层实现

1. 在ISupplierBiz中添加方法：



1. 在SupplierBiz中添加方法的实现：

|  |
| --- |
| /\*\*  \* 导出到excel文件  \* **@param** os 输出流  \* **@param** t1 查询条件  \*/  **public** **void** export(OutputStream os, Supplier t1){  // 根据查询条件获取供应商/客户列表  List<Supplier> supplierList = **super**.getList(t1, **null**, **null**);  // 创建excel工作簿  HSSFWorkbook wk = **new** HSSFWorkbook();  HSSFSheet sheet = **null**;  // 根据查询条件中的类型来创建相应名称的工作表  **if**("1".equals(t1.getType())){  sheet = wk.createSheet("供应商");  }  **if**("2".equals(t1.getType())){  sheet = wk.createSheet("客户");  }  // 写入表头  HSSFRow row = sheet.createRow(0);  // 定义好每一列的标题  String[] headerNames = {"名称","地址","联系人","电话","Email"};  // 指定每一列的宽度  **int**[] columnWidths = {4000,8000,2000,3000,8000};  HSSFCell cell = **null**;  **for**(**int** i = 0; i < headerNames.length; i++){  cell = row.createCell(i);  cell.setCellValue(headerNames[i]);  sheet.setColumnWidth(i, columnWidths[i]);  }  // 写入内容  **int** i = 1;  **for**(Supplier supplier : supplierList){  row = sheet.createRow(i);  //必须按照hderarNames的顺序来  row.createCell(0).setCellValue(supplier.getName());//名称  row.createCell(1).setCellValue(supplier.getAddress());//地址  row.createCell(2).setCellValue(supplier.getContact());//联系人  row.createCell(3).setCellValue(supplier.getTele());//联系电话  row.createCell(4).setCellValue(supplier.getEmail());//邮件地址  i++;  }  **try** {  // 写入到输出流中  wk.write(os);  } **catch** (IOException e) {  e.printStackTrace();  }**finally**{  **try** {  // 关闭工作簿  wk.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  } |

* + 1. Action层实现

修改SupplierAction，添加export方法：

|  |
| --- |
| /\*\*  \* 导出excel文件  \*/  **public** **void** export(){  String filename = "";  // 根据类型生成文件名  **if**("1".equals(getT1().getType())){  filename = "供应商.xls";  }  **if**("2".equals(getT1().getType())){  filename = "客户.xls";  }  HttpServletResponse response = ServletActionContext.*getResponse*();  **try** {  response.setHeader("Content-Disposition", "attachment;filename=" +  **new** String(filename.getBytes(),"ISO-8859-1"));//中文名称进行转码  //调用导出业务  supplierBiz.export(response.getOutputStream(), getT1());  } **catch** (UnsupportedEncodingException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }  } |

* 1. 页面代码
     1. 引入download.js文件

把资料中的download.js文件复制到webapp/ui目录下，并在supplier.html中引入



* + 1. 添加导出按钮

在crud.js中的grid的工具栏中添加按钮，代码如下



1. 采购销售订单导出
   1. 需求分析



* 1. POI样式设置

我们可以在示例工程中来完成这些导出的采购订单的模板，将来用到ERP系统里时，可直接把代码直接复制过去就可以用了。

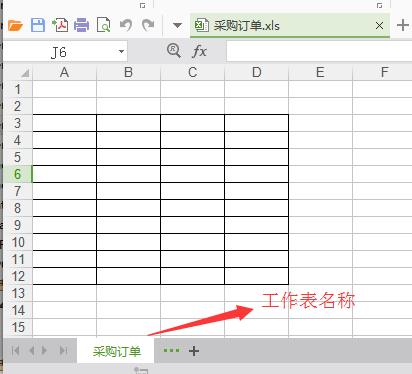
* + 1. 画边框线

边框线是属于样式范畴，而样式的作用范围是整个工作簿(包括所有的工作表)，因此我们可以工作簿来创建样式，再给样式设置边框线

创建CreateOrderXls类，代码如下

|  |
| --- |
| **package** cn.itcast.demo;  **import** java.io.FileOutputStream;  **import** org.apache.poi.hssf.usermodel.HSSFCell;  **import** org.apache.poi.hssf.usermodel.HSSFCellStyle;  **import** org.apache.poi.hssf.usermodel.HSSFRow;  **import** org.apache.poi.hssf.usermodel.HSSFSheet;  **import** org.apache.poi.hssf.usermodel.HSSFWorkbook;  /\*\*  \* 创建订单模板  \*/  **public** **class** CreateOrderXls {  **public** **static** **void** main(String[] args) **throws** Exception{  // 工作簿  HSSFWorkbook book = **new** HSSFWorkbook();  //　工作表  HSSFSheet sheet = book.createSheet("采购订单");  // 内容样式  HSSFCellStyle style\_content = book.createCellStyle();  style\_content.setBorderBottom(HSSFCellStyle.***BORDER\_THIN***);//下边框  style\_content.setBorderTop(HSSFCellStyle.***BORDER\_THIN***);//上边框  style\_content.setBorderLeft(HSSFCellStyle.***BORDER\_THIN***);//左边框  style\_content.setBorderRight(HSSFCellStyle.***BORDER\_THIN***);//右边框  // 根据导出的订单样本创建10行4列  **for**(**int** i = 2; i < 12; i++){  HSSFRow row = sheet.createRow(i);//创建订单内容中的行  **for**(**int** j = 0; j < 4; j++){  HSSFCell cell = row.createCell(j);//创建订单内容中的单元格  cell.setCellStyle(style\_content);//设置单元格的样式  }  }  //保存工作簿到本地目录  book.write(**new** FileOutputStream("c:\\采购订单.xls"));  book.close();  }  } |

打开结果文件：

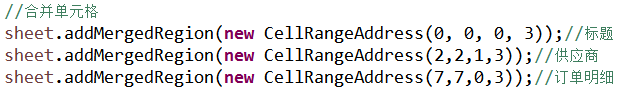


* + 1. 合并单元格

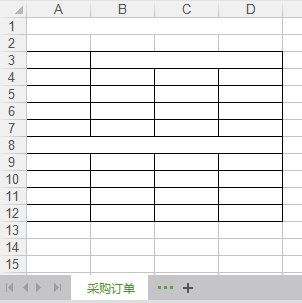
找出我们需要合并哪些单元格：

1. 第一行是显示订单的标题：比如显示为 “采购订单”，需要合并A、B、C、D列。
2. 第3行的B、C、D列，需要合并成一个单元格，用来显示供应商的名称
3. 第8行，需要合并成一个单元格，用来显示“订单详情”

在保存工作簿前添加如下代码：



结果：

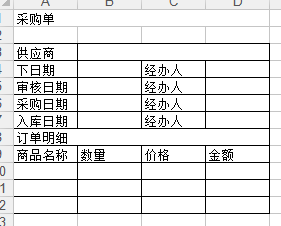


* + 1. 设置单元格的值

把采购订单里那些固定的显示文本给填上。在合并单元格代码后添加：



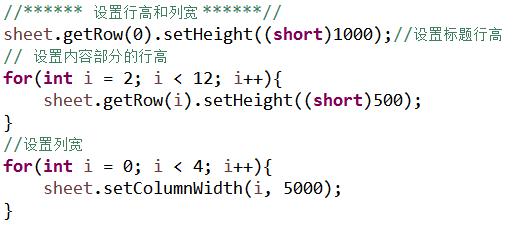
结果如下：



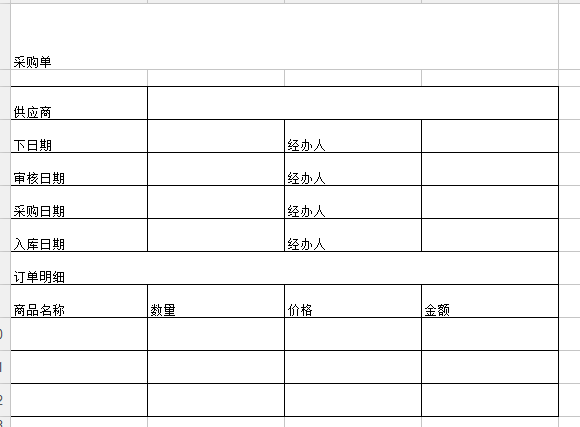
* + 1. 设置行高和列宽

输出的结果跟我们要的格式还有差距，需要调整一下行高和列宽

代码如下：



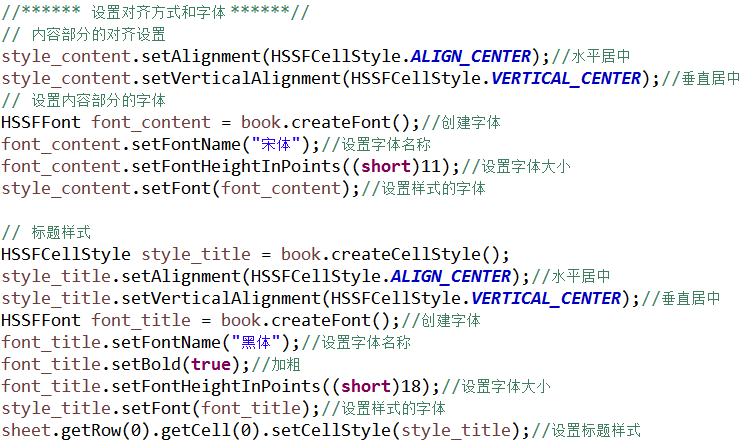
结果：



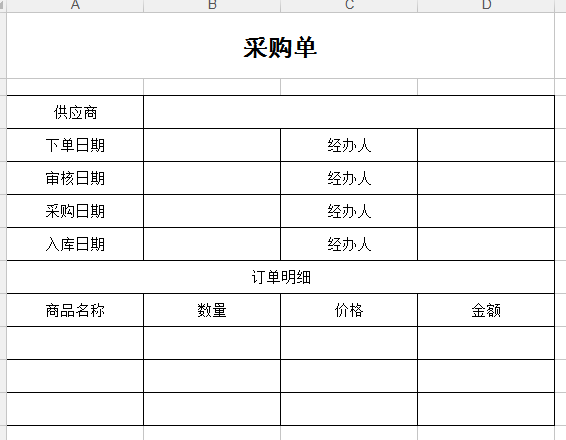
* + 1. 设置表格的对齐方式和字体

现在看到对齐方式全都是靠左对齐，需要改成居中，字体的大小也要设置一下

代码如下：



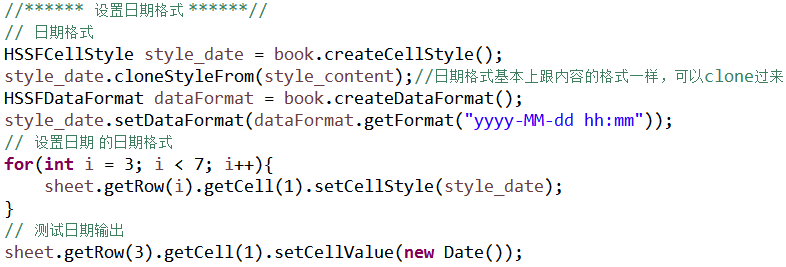
结果：



* + 1. 设置日期格式

订单表内容中有下单、审核、采购、入库日期。我们也需要设置一下日期格式的显示

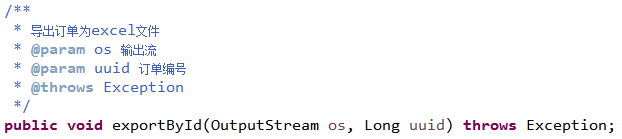
代码如下：



测试结果：



* 1. 后端代码实现
     1. 在IOrdersBiz接口中添加方法exportById

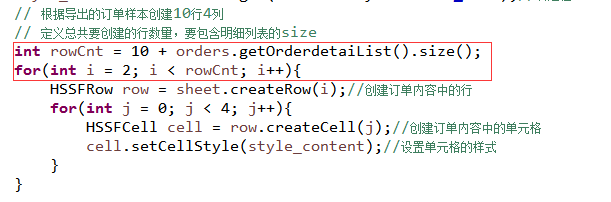


* + 1. 在OrdersBiz中实现方法

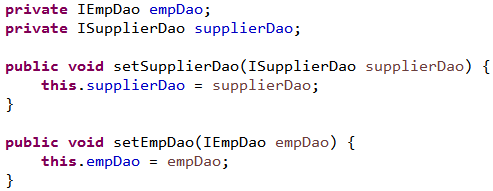
1. 将示例代码中的创建模板的代码复制过来，删除测试日期

|  |
| --- |
| // 工作簿  HSSFWorkbook book = **new** HSSFWorkbook();  //　工作表  HSSFSheet sheet = book.createSheet("采购订单");  // 内容样式  HSSFCellStyle style\_content = book.createCellStyle();  style\_content.setBorderBottom(HSSFCellStyle.***BORDER\_THIN***);//下边框  style\_content.setBorderTop(HSSFCellStyle.***BORDER\_THIN***);//上边框  style\_content.setBorderLeft(HSSFCellStyle.***BORDER\_THIN***);//左边框  style\_content.setBorderRight(HSSFCellStyle.***BORDER\_THIN***);//右边框  // 根据导出的订单样本创建10行4列  **for**(**int** i = 2; i < 12; i++){  HSSFRow row = sheet.createRow(i);//创建订单内容中的行  **for**(**int** j = 0; j < 4; j++){  HSSFCell cell = row.createCell(j);//创建订单内容中的单元格  cell.setCellStyle(style\_content);//设置单元格的样式  }  }  //合并单元格  sheet.addMergedRegion(**new** CellRangeAddress(0, 0, 0, 3));//标题  sheet.addMergedRegion(**new** CellRangeAddress(2,2,1,3));//供应商  sheet.addMergedRegion(**new** CellRangeAddress(7,7,0,3));//订单明细  //\*\*\*\*\*\* 设置固定文本内容 \*\*\*\*\*\*//  //设置标题内容, 注意：单元格必须创建后才能设置。  sheet.createRow(0).createCell(0).setCellValue("采购单");//设置标题内容  sheet.getRow(2).getCell(0).setCellValue("供应商");//设置供应商文本  //已经创建过的row/cell，则通过sheet.getRow.getCell方式获取  sheet.getRow(3).getCell(0).setCellValue("下单日期");  sheet.getRow(3).getCell(2).setCellValue("经办人");  sheet.getRow(4).getCell(0).setCellValue("审核日期");  sheet.getRow(4).getCell(2).setCellValue("经办人");  sheet.getRow(5).getCell(0).setCellValue("采购日期");  sheet.getRow(5).getCell(2).setCellValue("经办人");  sheet.getRow(6).getCell(0).setCellValue("入库日期");  sheet.getRow(6).getCell(2).setCellValue("经办人");  sheet.getRow(7).getCell(0).setCellValue("订单明细");  sheet.getRow(8).getCell(0).setCellValue("商品名称");  sheet.getRow(8).getCell(1).setCellValue("数量");  sheet.getRow(8).getCell(2).setCellValue("价格");  sheet.getRow(8).getCell(3).setCellValue("金额");  //\*\*\*\*\*\* 设置行高和列宽 \*\*\*\*\*\*//  sheet.getRow(0).setHeight((**short**)1000);//设置标题行高  // 设置内容部分的行高  **for**(**int** i = 2; i < 12; i++){  sheet.getRow(i).setHeight((**short**)500);  }  //设置列宽  **for**(**int** i = 0; i < 4; i++){  sheet.setColumnWidth(i, 5000);  }  //\*\*\*\*\*\* 设置对齐方式和字体 \*\*\*\*\*\*//  // 内容部分的对齐设置  style\_content.setAlignment(HSSFCellStyle.***ALIGN\_CENTER***);//水平居中  style\_content.setVerticalAlignment(HSSFCellStyle.***VERTICAL\_CENTER***);//垂直居中  // 设置内容部分的字体  HSSFFont font\_content = book.createFont();//创建字体  font\_content.setFontName("宋体");//设置字体名称  font\_content.setFontHeightInPoints((**short**)11);//设置字体大小  style\_content.setFont(font\_content);//设置样式的字体  // 标题样式  HSSFCellStyle style\_title = book.createCellStyle();  style\_title.setAlignment(HSSFCellStyle.***ALIGN\_CENTER***);//水平居中  style\_title.setVerticalAlignment(HSSFCellStyle.***VERTICAL\_CENTER***);//垂直居中  HSSFFont font\_title = book.createFont();//创建字体  font\_title.setFontName("黑体");//设置字体名称  font\_title.setBold(**true**);//加粗  font\_title.setFontHeightInPoints((**short**)18);//设置字体大小  style\_title.setFont(font\_title);//设置样式的字体  sheet.getRow(0).getCell(0).setCellStyle(style\_title);//设置标题样式  //\*\*\*\*\*\* 设置日期格式 \*\*\*\*\*\*//  // 日期格式  HSSFCellStyle style\_date = book.createCellStyle();  style\_date.cloneStyleFrom(style\_content);//日期格式基本上跟内容的格式一样，可以clone过来  HSSFDataFormat dataFormat = book.createDataFormat();  style\_date.setDataFormat(dataFormat.getFormat("yyyy-MM-dd hh:mm"));  // 设置日期 的日期格式  **for**(**int** i = 3; i < 7; i++){  sheet.getRow(i).getCell(1).setCellStyle(style\_date);  }  //保存工作簿到本地目录  book.write(**new** FileOutputStream("c:\\采购订单.xls"));  book.close(); |

1. 修改创建内容部分的代码，添加rowCnt定义，代表内容部分需要创建的总行数



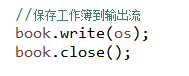
1. 注入EmpDao和SupplierDao，订单中只保留经办人的编号和供应商的编号，需要把它们转成对应的名称



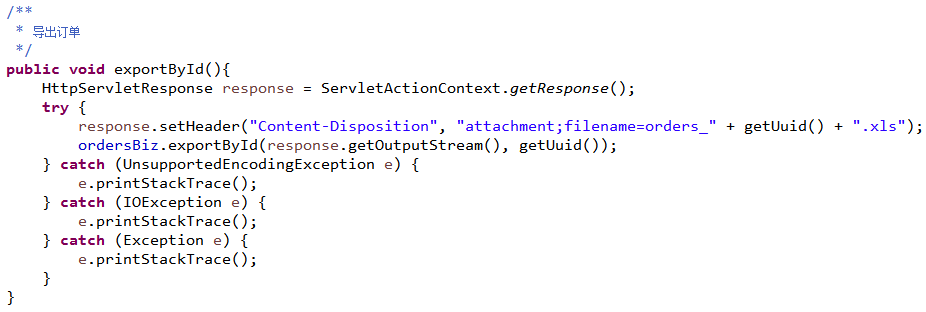
1. 在代码的后面添加：

|  |
| --- |
| //设置供应商的值  sheet.getRow(2).getCell(1).setCellValue(supplierDao.get(orders.getSupplieruuid()).getName());  //\*\*\*\*\*\* 设置日期内容 \*\*\*\*\*\*//  **if**(**null** != orders.getCreatetime()){  sheet.getRow(3).getCell(1).setCellValue(orders.getCreatetime());  }  **if**(**null** != orders.getChecktime()){  sheet.getRow(4).getCell(1).setCellValue(orders.getChecktime());  }  **if**(**null** != orders.getStarttime()){  sheet.getRow(5).getCell(1).setCellValue(orders.getStarttime());  }  **if**(**null** != orders.getEndtime()){  sheet.getRow(6).getCell(1).setCellValue(orders.getEndtime());  }  //\*\*\*\*\*\* 设置经办人的值 \*\*\*\*\*\*//  **if**(**null** != orders.getCreater()){//下单员  sheet.getRow(3).getCell(3).setCellValue(empDao.get(orders.getCreater()).getName());  }  **if**(**null** != orders.getChecker()){//审核员  sheet.getRow(4).getCell(3).setCellValue(empDao.get(orders.getChecker()).getName());  }  **if**(**null** != orders.getStarter()){//采购员  sheet.getRow(5).getCell(3).setCellValue(empDao.get(orders.getStarter()).getName());  }  **if**(**null** != orders.getEnder()){ //库管员  sheet.getRow(6).getCell(3).setCellValue(empDao.get(orders.getEnder()).getName());  }  //\*\*\*\*\*\* 订单明细 \*\*\*\*\*\*//  **int** rowIndex = 9;  HSSFRow row = **null**;  **for**(Orderdetail od : orders.getOrderdetaiList()){  row = sheet.getRow(rowIndex);  row.getCell(0).setCellValue(od.getGoodsname());  row.getCell(1).setCellValue(od.getNum());  row.getCell(2).setCellValue(od.getPrice());  row.getCell(3).setCellValue(od.getMoney());  rowIndex++;  }  //\*\*\*\*\*\* 合计 \*\*\*\*\*\*//  sheet.getRow(rowIndex).getCell(0).setCellValue("合计");  sheet.getRow(rowIndex).getCell(3).setCellValue(orders.getTotalmoney()); |

1. 修改 输出给输出流：

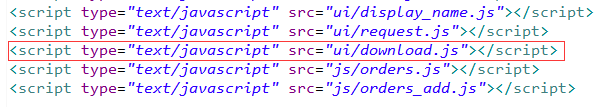


1. 修改OrdersAction，添加exportById方法



* 1. 前端代码实现
     1. 引入download.js

在orders.html中引入download.js



* + 1. 修改orders.js

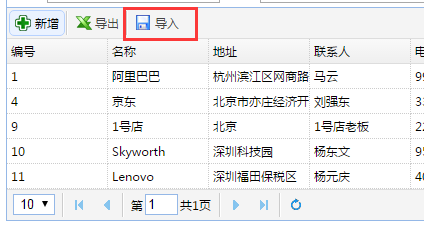
修改弹出订单详情窗口代码：



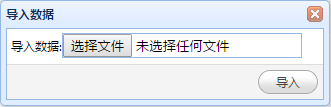
1. 供应商或客户数据导入
   1. 需求分析

我们需要将批量的供应商（客户）信息导入到系统里，因此我们需要实现导入功能

如下图：点击导入按钮



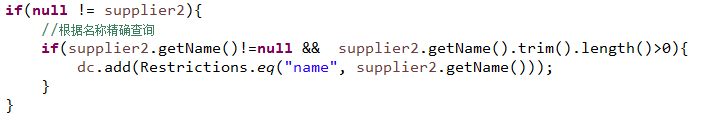
弹出导入数据对话框：



选择excel文件后点击“导入”按钮，把数据导入到系统中，成功后刷新表格并关闭“导入数据”窗口

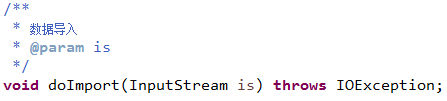
* 1. 后端代码实现
     1. Dao层编写

修改SupplierDao，添加根据名称查询的条件：



* + 1. 业务逻辑层编写

1. 在ISupplierBiz中添加方法



1. 在SupplierBiz中添加实现



* + 1. Action编写

1. 在SupplierAction中添加上传文件的支持：

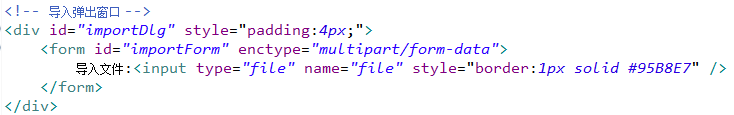


1. 添加upload方法：



* 1. 前端代码

1. 添加导入窗口



1. 修改crud.js，添加导入按钮：



1. 添加导入窗口的初始化与提交



1. ProcessData:

要求为Boolean类型的参数，默认为true。默认情况下，发送的数据将被转换为对象（从技术角度来讲关非字符串）以配合默认内容类型”application/x-www-form-urlencoded”。如果要发送DOM树信息或者其它不希望转换的信息，请设置为false。

1. ContentType:

要求为String类型的参数，当发送信息至服务器时，内容编码类型默认为”application/x-www-form-urlencoded”。该默认值适合大多数应用场合