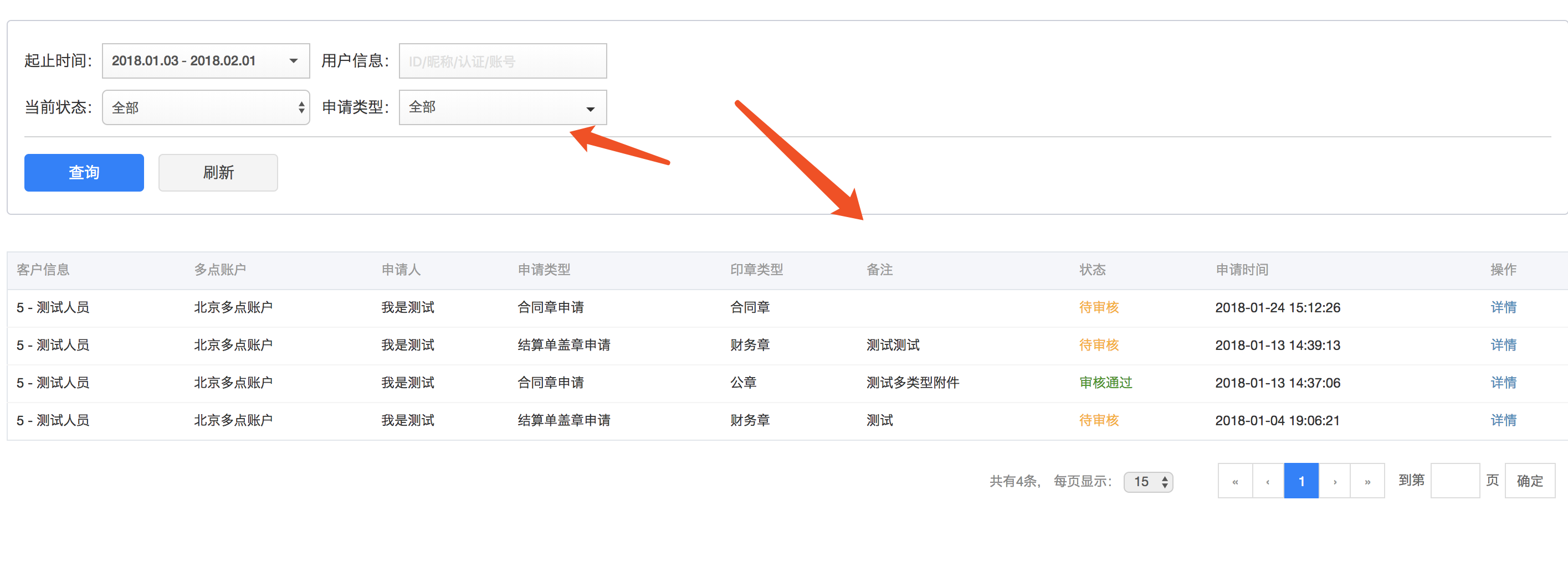
1、Excel使用工具类导出数据

# 1、js发起导出表单数据请求（参数全部为查询条件，其实就是讲查询条件获取的数据全部导出到excel'）



$scope.export = function(){  
  
 if($scope.query.userParam=undefined){  
 $scope.query.userParam = '';  
 }  
 ***window***.open(***ctx*** + '/verify/signet/export?status='+$scope.query.status  
 +'&startDate='+$scope.query.startDate  
 +'&endDate='+$scope.query.endDate  
 +'&userParam='+$scope.query.userParam  
 +'&page='+$scope.query.page  
 +'&pageSize='+$scope.query.pageSize  
 );  
};

# 2、contrller接收

## 1、查出的数据和分页查出的数据时一样的。

## 2、excel中不能含有java类型的数据Date，所以要将他转化为字符串

## 3、Excel的文件名为 开始时间-结束时间 中的一种



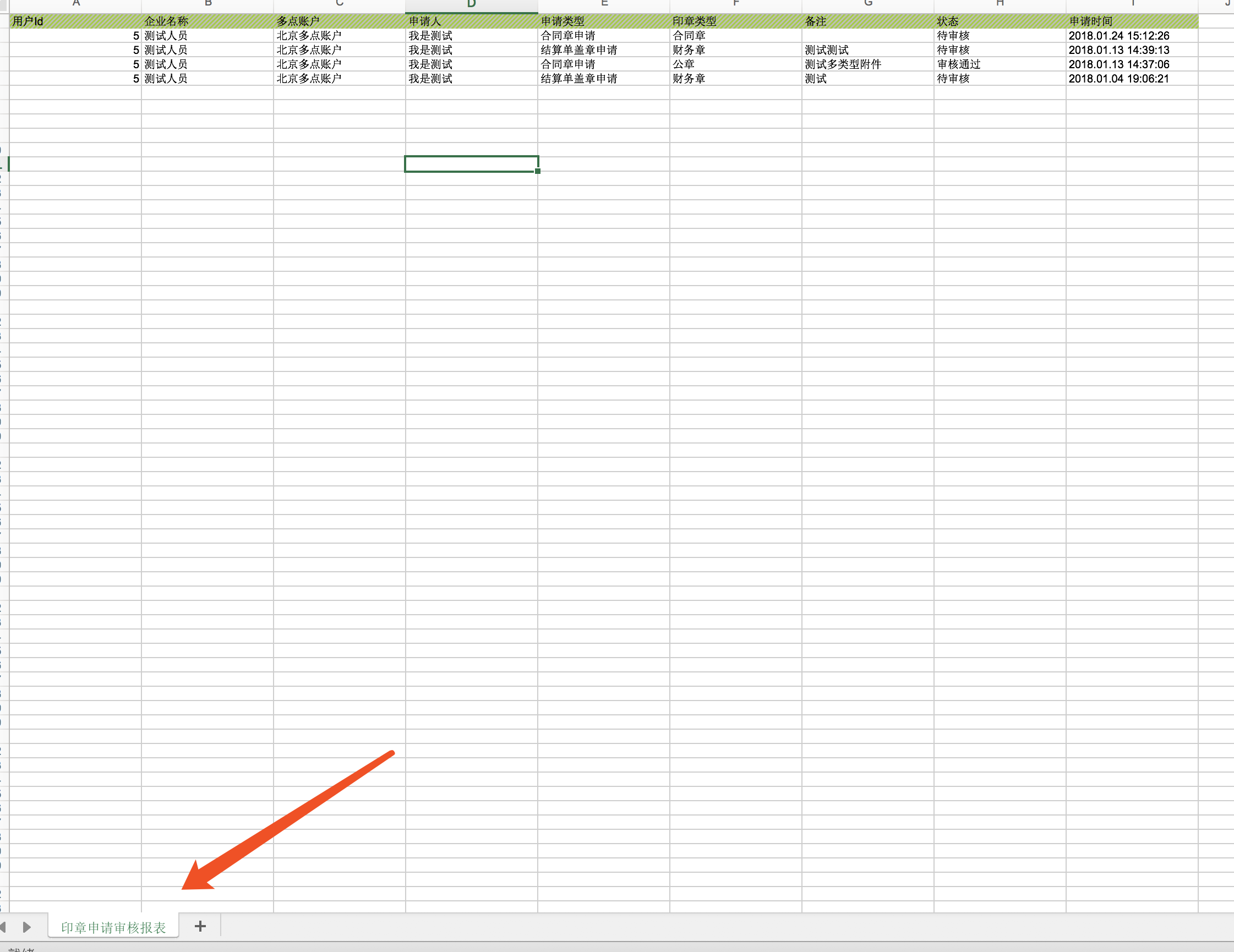
## 4、excel的导出准备

### 1、工作簿的名称

## 2、列的名称，注意和上面的浏览器中看到的一一对应（我这里的用户id和企业名称是两个字段，所以分开放了）

### 3、列在实体中的字段名称，

## 4、列的宽度



/\*\*  
 \* 导出数据  
 \* @param page  
 \* @param pageSize  
 \* @param query  
 \* @param response  
 \*/  
@RequestMapping("export")  
public void exportData(@RequestParam(defaultValue = "0")Integer page,

@RequestParam(defaultValue = "15")Integer pageSize,  
 SignetQuery query,

HttpServletResponse response) throws IOException {  
  
 if (page == null) page = 0;  
 Pageable pageable = new PageRequest(page, pageSize);  
 SysAdminUser user = AppSessionHelper.getSessionUser();  
 List<SignetApplyRecord> signetApplyRecords = verifySignetService.getVerifySignetData(user.getId(), pageable, query);  
  
 for (SignetApplyRecord signetApplyRecord : signetApplyRecords) {  
 signetApplyRecord.setCtime(DateHelper.convertDate2String(signetApplyRecord.getCdate(), "yyyy.MM.dd HH:mm:ss"));  
 }  
  
 String fileName = "";

Date startDate = query.getStartDate();  
 Date endDate = query.getEndDate();  
 if (startDate != null && endDate != null) {  
 fileName = DateHelper.convertDate2String(startDate, DateHelper.***YYYY\_MM\_DD***) + "至"  
 + DateHelper.convertDate2String(endDate, DateHelper.***YYYY\_MM\_DD***);  
 } else if (startDate != null) {  
 fileName = DateHelper.convertDate2String(startDate, DateHelper.***YYYY\_MM\_DD***) + "之后";  
 } else if (endDate != null) {  
 fileName = DateHelper.convertDate2String(endDate, DateHelper.***YYYY\_MM\_DD***) + "之前";  
 } else {  
 fileName = "全部时间";  
 }  
  
 try {  
 String[] columnNames = new String[]{"用户Id", "企业名称","多点账户","申请人","申请类型","印章类型","备注","状态","申请时间"};  
 String[] dbColumnNames = new String[]{"userId", "userNickName", "remittanceAccountDesc", "applyAdmName", "taskTypeDesc", "typeDesc", "remark","statusDesc","ctime"};  
 Integer[] columnWidths = new Integer[]{20, 20, 20, 20, 20, 20, 20, 20, 20};  
  
 new SimpleExcelExporter(fileName + ".xls", "印章申请审核报表", signetApplyRecords)  
 .withHeaders(columnNames)  
 .withContentFields(dbColumnNames)  
 .withColumnWidths(columnWidths)  
 .exportTo(response);  
 } catch (Exception e) {  
 logger.error(e.getMessage(), e);  
 response.getWriter().write("导出失败,请联系管理员:" + e.getMessage());  
 }  
}

# 3、excel工具类

package com.duodian.admore.admin.module.report.helper;  
  
import com.duodian.admore.core.helper.HeaderHelper;  
import org.apache.commons.io.IOUtils;  
import org.apache.poi.hssf.usermodel.\*;  
import org.apache.poi.hssf.util.HSSFColor;  
import org.apache.poi.ss.usermodel.\*;  
import org.springframework.beans.BeanWrapper;  
import org.springframework.beans.BeanWrapperImpl;  
  
import javax.servlet.http.HttpServletResponse;  
import java.io.UnsupportedEncodingException;  
import java.math.BigDecimal;  
import java.net.URLEncoder;  
import java.util.List;  
  
/\*\*  
 \* 类描述： 简单excel导出类  
 \* 创建人： j.sh  
 \* 创建时间： 07/02/2017  
 \* version：1.0.0  
 \*/  
public class SimpleExcelExporter {  
  
 private static final int ***WORD\_UNIT*** = 256;  
  
 private String[] headers;  
 private String[] contentFields;  
 private Integer[] columnWidths;  
  
 private String fileName;  
 private String sheetName;  
 private List content;  
  
 public SimpleExcelExporter(String fileName, String sheetName, List content) {  
 this.fileName = fileName;  
 this.sheetName = sheetName;  
 this.content = content;  
 }  
  
 public SimpleExcelExporter withHeaders(String[] headers){  
 this.headers = headers;  
 return this;  
 }  
  
 public SimpleExcelExporter withContentFields(String[] contentFields){  
 this.contentFields = contentFields;  
 return this;  
 }  
  
 public SimpleExcelExporter withColumnWidths(Integer[] columnWidths){  
 this.columnWidths = columnWidths;  
 return this;  
 }  
  
 public void exportTo(HttpServletResponse response) {  
 if (headers == null || headers.length == 0){  
 throw new IllegalArgumentException("headers not set yet");  
 }  
 if (contentFields == null || contentFields.length == 0){  
 throw new IllegalArgumentException("contentFields not set yet");  
 }  
  
 HSSFWorkbook workbook = null;  
 try {  
 response.setContentType("APPLICATION/OCTET-STREAM");  
  
 HeaderHelper.setResponseHeader(response,fileName);  
// response.setHeader("Content-Disposition", "attachment; filename=" + URLEncoder.encode(fileName, "UTF-8"));  
  
 workbook = new HSSFWorkbook();  
 HSSFSheet sheet = workbook.createSheet(sheetName);  
  
 this.setColumnWidth(sheet);  
 this.createTitle(workbook,sheet);  
 this.createContent(workbook,sheet, content);  
  
 workbook.write(response.getOutputStream());  
 } catch (Exception e) {  
 throw new RuntimeException(e.getMessage(), e);  
 } finally {  
 IOUtils.closeQuietly(workbook);  
 }  
 }  
  
 private HSSFCellStyle createIntegerCellStyle(HSSFWorkbook workbook) {  
 DataFormat format = workbook.createDataFormat();  
 HSSFCellStyle cellStyle = workbook.createCellStyle();  
 cellStyle.setDataFormat(format.getFormat("#,##0"));  
 cellStyle.setAlignment(HorizontalAlignment.***RIGHT***);  
 return cellStyle;  
 }  
  
 private HSSFCellStyle createFloatCellStyle(HSSFWorkbook workbook) {  
 DataFormat format = workbook.createDataFormat();  
 HSSFCellStyle cellStyle = workbook.createCellStyle();  
 cellStyle.setDataFormat(format.getFormat("#,##0.00"));  
 cellStyle.setAlignment(HorizontalAlignment.***RIGHT***);  
 return cellStyle;  
 }  
  
 private HSSFCellStyle createTitleCellStyle(HSSFWorkbook workbook) {  
 HSSFCellStyle cellStyle = workbook.createCellStyle();  
 Font font = workbook.createFont();  
 cellStyle.setFont(font);  
  
 cellStyle.setFillForegroundColor(HSSFColor.LIME.***index***);  
 cellStyle.setFillPattern(FillPatternType.***THICK\_FORWARD\_DIAG***);  
 cellStyle.setFillBackgroundColor(IndexedColors.***GREY\_25\_PERCENT***.index);  
 return cellStyle;  
 }  
  
 private void setColumnWidth(HSSFSheet sheet) {  
 if (columnWidths == null || columnWidths.length == 0){  
 for (int i = 0; i < headers.length; i++) {  
 sheet.autoSizeColumn(i);  
 }  
 } else {  
 for (int i = 0; i < columnWidths.length; i++) {  
 sheet.setColumnWidth(i, columnWidths[i] \* ***WORD\_UNIT***);  
 }  
 }  
 }  
  
 private void createTitle(HSSFWorkbook workbook,HSSFSheet sheet) {  
 HSSFCellStyle titleStyle = this.createTitleCellStyle(workbook);  
  
 HSSFRow row = sheet.createRow(0);  
  
 for (int i = 0; i < headers.length; i++) {  
 HSSFCell cell = row.createCell(i);  
 cell.setCellValue(headers[i]);  
 cell.setCellStyle(titleStyle);  
 }  
 }  
  
 private void createContent(HSSFWorkbook workbook,HSSFSheet sheet, List content) {  
 HSSFCellStyle integerStyle = this.createIntegerCellStyle(workbook);  
 HSSFCellStyle floatStyle = this.createFloatCellStyle(workbook);  
  
 for (int i = 0; i < content.size(); i++) {  
 HSSFRow row = sheet.createRow(i + 1); //排除标题行，从1开始  
 BeanWrapper wrapper = new BeanWrapperImpl(content.get(i));  
  
 for (int j = 0; j < contentFields.length; j++) {  
 HSSFCell cell = row.createCell(j);  
  
 Object value = wrapper.getPropertyValue(contentFields[j]);  
 if (value == null){  
 value = "";  
 }  
  
 this.setCellValue(cell,value,integerStyle,floatStyle);  
 }  
 }  
 }  
  
 private void setCellValue(HSSFCell cell,Object value,HSSFCellStyle integerStyle,HSSFCellStyle floatStyle){  
 if (value instanceof String){  
 cell.setCellValue(value.toString());  
 } else if (value instanceof Integer || value instanceof Long){  
 cell.setCellValue(Long.parseLong(value.toString()));  
 cell.setCellStyle(integerStyle);  
 } else if (value instanceof Float || value instanceof Double || value instanceof BigDecimal){  
 cell.setCellValue(Double.parseDouble(value.toString()));  
 cell.setCellStyle(floatStyle);  
 }  
 }  
  
}

# 4、filename文件名输出制作

package com.duodian.admore.core.helper;  
  
import javax.servlet.http.HttpServletResponse;  
import java.io.UnsupportedEncodingException;  
import java.net.URLEncoder;  
  
/\*\*  
 \* 类描述：  
 \* 创建人：lishiwei  
 \* 创建时间：2017/8/23  
 \* 修改人：  
 \* 修改时间：  
 \* 修改备注：  
 \*  
 \* @version 1.0.0  
 \*/  
public class HeaderHelper {  
  
 /\*\*  
 \* <pre>  
 \* 浏览器下载文件时需要在服务端给出下载的文件名，当文件名是ASCII字符时没有问题  
 \* 当文件名有非ASCII字符时就有可能出现乱码  
 \*  
 \* 这里的实现方式参考这篇文章  
 \* http://blog.robotshell.org/2012/deal-with-http-header-encoding-for-file-download/  
 \*  
 \* 最终设置的response header是这样:  
 \*  
 \* Content-Disposition: attachment;  
 \* filename="encoded\_text";  
 \* filename\*=utf-8''encoded\_text  
 \*  
 \* 其中encoded\_text是经过RFC 3986的“百分号URL编码”规则处理过的文件名  
 \* </pre>  
 \* @param response  
 \* @param filename  
 \* @return  
 \*/  
 public static void setResponseHeader(HttpServletResponse response, String filename) throws UnsupportedEncodingException {  
 String headerValue = "attachment;";  
 headerValue += " filename=\"" + encodeURIComponent(filename) +"\";";  
 headerValue += " filename\*=utf-8''" + encodeURIComponent(filename);  
 response.setHeader("Content-Disposition", headerValue);  
 }  
  
 /\*\*  
 \* <pre>  
 \* 符合 RFC 3986 标准的“百分号URL编码”  
 \* 在这个方法里，空格会被编码成%20，而不是+  
 \* 和浏览器的encodeURIComponent行为一致  
 \* </pre>  
 \* @param value  
 \* @return  
 \*/  
 private static String encodeURIComponent(String value) throws UnsupportedEncodingException {  
 return URLEncoder.encode(value, "UTF-8").replaceAll("\\+", "%20");  
 }  
}