

# 【(轻量级ExcelUtil)基于JAVA反射的Excel导入导出万能工具类】和【基于Sring Boot及Apache POI的Excel导入导出示例】。



三汪 (/u/deee856014f6) [+ 关注](#)

2017.09.19 15:28\* 字数 551 阅读 98 评论 0 喜欢 10

(/u/deee856014f6)

## 前言

前几天发了一篇文章，提供了基于最新POI版本的Excel导出示例，提供了网上各个现有代码版本的deprecated警告的解决方案。今天来提供一个万能的导入导出工具类模板以及相应示例。

工具类的思路和代码骨架folk自素剑步轻尘 (<https://gitee.com/likeixuan0/excelutil>)，我个人基于实际需求对代码进行了重写和优化，并补充完善了代码注释。查看工具类代码已在开源中国上发布:获取代码戳[这里](https://gitee.com/wolfgymoy/excelutil) (<https://gitee.com/wolfgymoy/excelutil>)。使用模板只需要将ExcelUtil.java和StringUtil.java复制到你的项目中即可。

**【划重点：同步支持xls和xlsx版本。实现导入或者导出都只需要两步操作。】**

### 读取Excel调用步骤:

1.定义需要读取的表头字段和表头对应的属性字段

```
String keyValue = "手机名称:phoneName,颜色:color,售价:price";
```

2.读取数据

```
List<PhoneModel> list = ExcelUtil.readExcel("test.xlsx", new  
FileInputStream("E://test.xlsx"), ExcelUtil.getMap(keyValue), "com.lkx.model.PhoneModel", 1);
```

readExcel参数说明:

```
/**  
 * readExcel:根据传进来的map集合读取Excel以及model读取Excel文件  
 *  
 * @author likeixuan,wolfgymoy  
 * @version 1.1 2017年9月18日  
 * @param fileName  
 *         Excel文件名  
 * @param inputStream 输入流  
 * @param map  
 *         表头和属性的Map集合,其中Map中Key为Excel列的名称, Value为反射类的属性  
 * @param classPath  
 *         需要映射的model的路径  
 * @param rowNumIndex  
 *         表头所在行数(从1开始,即第一行对应行数1)  
 * @return List<T> 读取到的数据集合  
 * @throws Exception  
 * @since JDK 1.7  
 */
```

### 导出Excel调用步骤

1.定义需要读取的表头字段和表头对应的属性字段

```
String keyValue = "手机名称:phoneName,颜色:color,售价:price";
```

2.导出

```
ExcelUtil.exportExcel("导出数据", new FileOutputStream("E://testOut.xls"),  
ExcelUtil.getMap(keyValue), list, "com.lkx.model.PhoneModel", null, null, null);
```

exportExcel参数说明:



```
/**
 *
 * <p>
 * Description:Excel导出<br />
 * </p>
 * @author likaixuan,wolfgy
 * @version 1.1 2017年9月18日
 * @param titleText 标题栏内容
 * @param out 输出流
 * @param map 表头和属性的Map集合,其中Map中Key为Excel列的名称,Value为反射类的属性
 * @param list 要输出的对象集合
 * @param classPath 需要映射的model的路径
 * @param titleStyle 标题栏样式。若为null则直接使用默认样式
 * @param headStyle 表头样式。若为null则直接使用默认样式
 * @param dataStyle 数据行样式。若为null则直接使用默认样式
 * @throws Exception
 * @since JDK 1.7
 * void
 */
```

下面简单粗暴上项目结构和代码。  
有什么意见、见解或疑惑，欢迎留言讨论。

## 项目结构

- util层**：ExcelUtil.java,StringUtil.java 实现导入导出Excel的工具类及其依赖的String工具类
- web层**：AssociationController.java 接口类
- service层**：StudentService.java 业务类
- domain层**：Student.java 学生实体

## 代码展示

ExcelUtil.java



```
/**
 * Project Name:excelutil
 * File Name:ExcelUtil.java
 * Package Name:com.lkx.util
 * Date:2017年6月7日上午9:44:58
 * Copyright (c) 2017~2020, likaixuan@test.com.cn All Rights Reserved.
 *
 */

package com.wolfgy.util;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.Serializable;
import java.lang.reflect.Method;
import java.math.BigDecimal;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.apache.commons.lang3.StringUtils;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFCellStyle;
import org.apache.poi.hssf.usermodel.HSSFFont;
import org.apache.poi.hssf.usermodel.HSSFRichTextString;
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.hssf.util.HSSFColor;
import org.apache.poi.hssf.util.HSSFColorPredefined;
import org.apache.poi.ss.usermodel.BorderStyle;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.DateUtil;
import org.apache.poi.ss.usermodel.FillPatternType;
import org.apache.poi.ss.usermodel.HorizontalAlignment;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.VerticalAlignment;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.util.CellRangeAddress;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeanUtils;

public class ExcelUtil implements Serializable{
    /**
     * serialVersionUID
     */
    private static final long serialVersionUID = 1L;

    private static final Logger LOGGER = LoggerFactory
        .getLogger(ExcelUtil.class);
    //设置Excel读取最大行数
    private static final int MAX_ROW = 20000;

    /**
     * getMap:(将传进来的表头和表头对应的属性存进Map集合, 表头字段为key,属性为value)
     *
     * @author likaixuan,wolfgy
     * @version 1.1 2017年9月18日
     * @param 把传进指定格式的字符串解析到Map中
     * 形如: String keyValue = "手机名称:phoneName,颜色:color,售价:price";
     * @return Map<String, String> 转换好的Map集合
     * @since JDK 1.7
     */
    public static Map<String, String> getMap(String keyValue) {
        Map<String, String> map = new HashMap<>();
        if (keyValue != null) {
            String[] str = keyValue.split(",");
            for (String element : str) {
                String[] str2 = element.split(":");
                map.put(str2[0], str2[1]);
            }
        }
        return map;
    }
}
```



```

}

/**
 * readExcel:根据传进来的map集合读取Excel以及model读取Excel文件
 *
 * @author likaixuan,wolfgu
 * @version 1.1 2017年9月18日
 * @param fileName
 *          Excel文件名
 * @param inputStream 输入流
 * @param map
 *          表头和属性的Map集合,其中Map中Key为Excel列的名称, Value为反射类的属性
 * @param classPath
 *          需要映射的model的路径
 * @param rowNumIndex
 *          表头所在行数(从1开始,即第一行对应行数1)
 * @return List<T> 读取到的数据集合
 * @throws Exception
 * @since JDK 1.7
 */
@SuppressWarnings({ "resource", "unchecked" })
public static <T> List<T> readExcel(String fileName, InputStream inputStream, Map<String,
String> classPath, int rowNumIndex) throws Exception {
    // 返回表头字段名和属性字段名Map集合中键的集合(Excel列的名称集合)
    Set<String> keySet = map.keySet();

    //反射用
    Class<?> demo = null;
    Object obj = null;
    List<Object> list = new ArrayList<Object>();
    demo = Class.forName(classPath);
    //获取文件名后判断文件类型
    String fileType = fileName.substring(fileName.lastIndexOf(".") + 1,
        fileName.length());

    //根据文件类型及文件输入流新建工作簿对象
    Workbook wb = null;
    if (fileType.equals("xls")) {
        wb = new HSSFWorkbook(inputStream);
    } else if (fileType.equals("xlsx")) {
        wb = new XSSFWorkbook(inputStream);
    } else {
        LOGGER.error("您输入的excel格式不正确");
        throw new Exception("您输入的excel格式不正确");
    }

    // 遍历每个Sheet表
    for (int sheetNum = 0; sheetNum < 1; sheetNum++) {
        // 表头成功读取标志位。当表头成功读取后, rowNum_x值为表头实际行数
        int rowNum_x = -1;
        // 存放每一个field字段对应所在的列的序号
        Map<String, Integer> cellmap = new HashMap<String, Integer>();
        // 存放所有的表头字段信息
        List<String> headlist = new ArrayList<>();
        // 获取当前Sheet表
        Sheet hssfSheet = wb.getSheetAt(sheetNum);

        //设置默认最大行数,当超出最大行数时返回异常
        if(hssfSheet != null && hssfSheet.getLastRowNum()>MAX_ROW){
            throw new Exception("Excel 数据超过20000行,请检查是否有空行,或分批导入");
        }

        // 遍历Excel中的每一行
        for (int rowNum = 0; rowNum <= hssfSheet.getLastRowNum(); rowNum++) {
            // 当表头成功读取标志位rowNum_x为-1时,说明还未开始读取数据。此时,如果传值指定读取
            if (rowNum_x == -1) {
                //判断指定行是否为空
                Row hssfRow = hssfSheet.getRow(rowNumIndex);
                if (hssfRow == null) {
                    throw new RuntimeException("指定的行为空,请检查");
                }
                //设置当前行为指定行
                rowNum = rowNumIndex - 1;
            }

            //获取当前行
            Row hssfRow = hssfSheet.getRow(rowNum);
            //当前行为空时,跳出本次循环进入下一行
            if (hssfRow == null) continue;

            //当前行数据为空时,跳出本次循环进入下一行
            boolean flag = false;
            for (int i = 0; i < hssfRow.getLastCellNum(); i++) {
                if (hssfRow.getCell(i) != null && !("").equals(hssfRow.getCell(i).toString()))
                    flag = true;
            }
            if (!flag) continue;

```



```

//获取表头内容
if (rowNum_x == -1) {
    // 循环列Cell
    for (int cellNum = 0; cellNum <= hssfRow
        .getLastCellNum(); cellNum++) {

        Cell hssfCell = hssfRow.getCell(cellNum);
        //当前cell为空时,跳出本次循环,进入下一列。
        if (hssfCell == null) {
            continue;
        }
        //获取当前cell的值(String类型)
        String tempCellValue = hssfSheet.getRow(rowNum)
            .getCell(cellNum).getStringCellValue();
        //去除空格,空格ASCII码为160
        tempCellValue = StringUtils.remove(tempCellValue,
            (char) 160);
        tempCellValue = tempCellValue.trim();
        //将表头内容放入集合
        headlist.add(tempCellValue);

        //遍历表头字段名和属性字段名Map集合中键的集合(Excel列的名称集合)
        Iterator<String> it = keySet.iterator();
        while (it.hasNext()) {
            Object key = it.next();
            if (StringUtils.isNotBlank(tempCellValue)
                && StringUtils.equals(tempCellValue,
                    key.toString())) {
                //将rowNum_x设为实际的表头行数
                rowNum_x = rowNum;
                //获取表头每一个field字段对应所在的列的序号
                cellmap.put(map.get(key).toString(), cellNum);
            }
        }
        //当rowNum_x为-1时,说明没有在表头找到对应的字段或者对应字段行上面含有不为空白的行
        if (rowNum_x == -1) {
            LOGGER.error("没有找到对应的字段或者对应字段行上面含有不为空白的行");
            throw new Exception("没有找到对应的字段或者对应字段行上面含有不为空白的行");
        }
    }
}

} else {
    //实例化反射类对象
    obj = demo.newInstance();
    //遍历并取出所需要的每个属性值
    Iterator<String> it = keySet.iterator();
    while (it.hasNext()) {
        //Excel列名
        Object key = it.next();
        //获取属性对应列数
        Integer cellNum_x = cellmap
            .get(map.get(key).toString());
        //当属性对应列为空时,结束本次循环,进入下次循环,继续获取其他属性值
        if (cellNum_x == null || hssfRow.getCell(cellNum_x) == null) {
            continue;
        }
        //得到属性名
        String attrName = map.get(key).toString();
        //得到属性类型
        Class<?> attrType = BeanUtils.findPropertyType(attrName,
            new Class[] { obj.getClass() });
        //得到属性值
        Cell cell = hssfRow.getCell(cellNum_x);
        Object val = getValue(cell, obj, attrName, attrType, rowNum, cellNum_x, key);
        setter(obj, attrName, val, attrType, rowNum, cellNum_x, key);
    }
    //将实例化好并设置完属性的对象放入要返回的list中
    list.add(obj);
}

}

}

wb.close();
inputStream.close();

return (List<T>) list;
}

/**
 *
 * <p>
 * Description:setter(反射set方法给属性赋值)<br />

```



```

* </p>
* @author likaixuan,wolfgy
* @version 1.1 2017年9月18日
* @param obj 反射类对象
* @param attrName 属性名
* @param attrValue 属性值
* @param attrType 属性类型
* @param row 当前数据在Excel中的具体行数
* @param column 当前数据在Excel中的具体列数
* @param key 当前数据对应的Excel列名
* @since JDK 1.7
* @throws Exception
* void
*/
public static void setter(Object obj, String attrName, Object attrValue,
    Class<?> attrType, int row, int column, Object key) throws Exception {
    try {
        //获取反射的方法名
        Method method = obj.getClass().getMethod(
            "set" + StringUtil.toUpperCaseFirstOne(attrName), attrType);
        //进行反射
        method.invoke(obj, attrValue);
    } catch (Exception e) {
        e.printStackTrace();
        LOGGER.error("第" + (row + 1) + "行 " + (column + 1) + "列 属性: " + key
            + " 赋值异常 " + e.getStackTrace());
        throw new Exception("第" + (row + 1) + "行 " + (column + 1) + "列 属性: "
            + key + " 赋值异常 ");
    }
}

/**
* <p>
* Description:getter(反射get方法得到属性值)<br />
* </p>
* @author likaixuan,wolfgy
* @version 1.1 2017年9月18日
* @param obj
* 反射类对象
* @param attrName
* 属性名
* @throws Exception
* @since JDK 1.7
*/
public static Object getter(Object obj, String attrName)
    throws Exception {
    try {
        //获取反射的方法名
        Method method = obj.getClass().getMethod("get" + StringUtil.toUpperCaseFirstOne(
            attrName), attrType);
        Object value = new Object();
        //进行反射并获取返回值
        value = method.invoke(obj);
        return value;
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

/**
*
* <p>
* Description:读取当前单元格的值<br />
* </p>
* @author likaixuan,wolfgy
* @version 1.1 2017年9月18日
* @param cell 单元格对象
* @param obj 反射类对象
* @param attrName 属性名
* @param attrType 属性类型
* @param row 当前数据在Excel中的具体行数
* @param col 当前数据在Excel中的具体列数
* @param key 当前数据对应的Excel列名
* @throws Exception
* @since JDK 1.7
* @return val 当前单元格的值
*/
public static Object getValue(Cell cell, Object obj, String attrName,
    Class<?> attrType, int row, int column, Object key) throws Exception {
    //新建当前单元格值对象
    Object val = null;
    //判断当前单元格数据类型并取值
    if (cell.getCellTypeEnum() == CellType.BOOLEAN) {
        val = cell.getBooleanCellValue();
    }
}

```



```

    } else if (cell.getCellTypeEnum() == CellType.NUMERIC) {
        if (DateUtil.isCellDateFormatted(cell)) {
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
            try {
                if (attrType == String.class) {
                    val = sdf.format(DateUtil
                        .getJavaDate(cell.getNumericCellValue()));
                } else {
                    val = StringUtil.dateConvertFormat(
                        sdf.format(DateUtil.getJavaDate(
                            cell.getNumericCellValue())));
                }
            } catch (ParseException e) {
                LOGGER.error("日期格式转换错误");
                throw new Exception("第" + (row + 1) + " 行 " + (column + 1)
                    + "列 属性: " + key + " 日期格式转换错误 ");
            }
        } else {
            if (attrType == String.class) {
                cell.setCellType(CellType.STRING);
                val = cell.getStringCellValue();
            } else if (attrType == BigDecimal.class) {
                val = new BigDecimal(cell.getNumericCellValue());
            } else if (attrType == Long.class) {
                val = (long) cell.getNumericCellValue();
            } else if (attrType == Double.class) {
                val = cell.getNumericCellValue();
            } else if (attrType == Float.class) {
                val = (float) cell.getNumericCellValue();
            } else if (attrType == int.class || attrType == Integer.class) {
                val = (int) cell.getNumericCellValue();
            } else if (attrType == Short.class) {
                val = (short) cell.getNumericCellValue();
            } else {
                val = cell.getNumericCellValue();
            }
        }
    }

    } else if (cell.getCellTypeEnum() == CellType.STRING) {
        val = cell.getStringCellValue();
    }

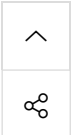
    return val;
}

/**
 *
 * <p>
 * Description:Excel导出<br />
 * </p>
 * @author likaixuan,wolfgy
 * @version 1.1 2017年9月18日
 * @param titleText 标题栏内容
 * @param out 输出流
 * @param map 表头和属性的Map集合,其中Map中Key为Excel列的名称, Value为反射类的属性
 * @param list 要输出的对象集合
 * @param classPath 需要映射的model的路径
 * @param titleStyle 标题栏样式。若为null则直接使用默认样式
 * @param headStyle 表头样式。若为null则直接使用默认样式
 * @param dataStyle 数据行样式。若为null则直接使用默认样式
 * @throws Exception
 * @since JDK 1.7
 * void
 */
public static void exportExcel(String titleText, OutputStream out, Map<String, String> map

    //创建单元格并设置单元格内容
    Set<String> keySet = map.keySet();// 返回键的集合
    Iterator<String> it = keySet.iterator();
    // 创建HSSFWorkbook对象(excel的文档对象)
    HSSFWorkbook workbook = new HSSFWorkbook();
    // 建立新的sheet对象 (excel的表单)
    HSSFSheet sheet = workbook.createSheet("数据导出");

    // 设置默认列宽为15
    sheet.setDefaultColumnWidth(15);
    // 合并标题栏单元格
    sheet.addMergedRegion(new CellRangeAddress(0, 0, 0, keySet.size() - 1));
    // 当传入的标题栏样式为空时, 创建默认标题栏样式
    if (titleStyle == null) {
        HSSFCellStyle style = workbook.createCellStyle();
        style.setFillForegroundColor(HSSFColor.Predefined.SKY_BLUE.getIndex());
        style.setFillPattern(FillPatternType.SOLID_FOREGROUND);
        style.setBorderBottom(BorderStyle.THIN);
        style.setBorderLeft(BorderStyle.THIN);
        style.setBorderRight(BorderStyle.THIN);
    }

```



```

        style.setBorderTop(BorderStyle.THIN);
        style.setAlignment(HorizontalAlignment.CENTER);
        style.setVerticalAlignment(VerticalAlignment.CENTER);
        HSSFFont font = workbook.createFont();
        font.setColor(HSSFColorPredefined.VIOLET.getIndex());
        font.setFontHeightInPoints((short) 18);
        style.setFont(font);
        titleStyle = style;
    }
    // 当传入的表头样式为空时, 创建默认表头样式
    if (headStyle == null) {
        HSSFCellStyle style2 = workbook.createCellStyle();
        style2.setFillForegroundColor(HSSFColorPredefined.GREEN.getIndex());
        style2.setFillPattern(FillPatternType.SOLID_FOREGROUND);
        style2.setBorderBottom(BorderStyle.THIN);
        style2.setBorderLeft(BorderStyle.THIN);
        style2.setBorderRight(BorderStyle.THIN);
        style2.setBorderTop(BorderStyle.THIN);
        style2.setAlignment(HorizontalAlignment.CENTER);
        style2.setVerticalAlignment(VerticalAlignment.CENTER);
        HSSFFont font2 = workbook.createFont();
        font2.setFontHeightInPoints((short) 12);
        style2.setFont(font2);
        headStyle = style2;
    }
    // 当传入的数据行样式为空时, 创建默认数据行样式
    if (dataStyle == null) {
        HSSFCellStyle style3 = workbook.createCellStyle();
        style3.setFillForegroundColor(HSSFColorPredefined.LIGHT_YELLOW.getIndex());
        style3.setFillPattern(FillPatternType.SOLID_FOREGROUND);
        style3.setBorderBottom(BorderStyle.THIN);
        style3.setBorderLeft(BorderStyle.THIN);
        style3.setBorderRight(BorderStyle.THIN);
        style3.setBorderTop(BorderStyle.THIN);
        style3.setAlignment(HorizontalAlignment.CENTER);
        style3.setVerticalAlignment(VerticalAlignment.CENTER);
        dataStyle = style3;
    }

    // 创建行、单元格对象
    HSSFRow row = null;
    HSSFCell cell = null;
    // 写入标题行
    row = sheet.createRow(0);
    row.setHeightInPoints(25);
    cell = row.createCell(0);
    cell.setCellStyle(titleStyle);
    HSSFRichTextString textTitle = new HSSFRichTextString(titleText);
    cell.setCellValue(textTitle);

    //写入表头
    row = sheet.createRow(1); // 参数为行索引(excel的行), 可以是0~65535之间的任何一个
    Map<String,String> attrMap = new HashMap<>();
    int index = 0;
    while(it.hasNext()){
        String key = it.next().toString();
        cell = row.createCell(index);
        cell.setCellValue(key);
        cell.setCellStyle(headStyle);
        attrMap.put(Integer.toString(index++), map.get(key).toString());
    }
    //写入数据行
    for(int i=2;i<list.size();i++){
        row = sheet.createRow(i);
        for(int j=0;j<map.size();j++){
            //调用getter获取要写入单元格的数据值
            Object value = getter(list.get(i), attrMap.get(Integer.toString(j)));
            cell = row.createCell(j);
            cell.setCellValue(value.toString());
            cell.setCellStyle(dataStyle);
        }
    }
}

// 输出Excel文件
try {
    workbook.write(out);
    out.flush();
    out.close();
    workbook.close();
    LOGGER.info("导出成功!");
} catch (IOException e) {
    LOGGER.info("IOException! 导出失败! ");
    e.printStackTrace();
}
}
}

```





```
}  
}
```

StringUtil.java



```
/**
 * Project Name:excelutil
 * File Name:StringUtil.java
 * Package Name:com.lkx.util
 * Date:2017年6月7日上午9:47:06
 * Copyright (c) 2017~2020, likaixuan@test.com.cn All Rights Reserved.
 *
 */

package com.wolfgy.util;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * ClassName:StringUtil
 * Function: TODO ADD FUNCTION.
 * Reason: TODO ADD REASON.
 * Date: 2017年6月7日 上午9:47:06
 * @author likaixuan
 * @version V1.0
 * @since JDK 1.7
 * @see
 */
public class StringUtil {

    /**
     * 首字母转小写
     *
     * @param s
     * @return
     */
    public static String toLowerCaseFirstOne(String s) {
        if (Character.isLowerCase(s.charAt(0))) {
            return s;
        } else {
            return (new StringBuilder())
                .append(Character.toLowerCase(s.charAt(0)))
                .append(s.substring(1)).toString();
        }
    }

    /**
     * 首字母转大写
     *
     * @param s
     * @return
     */
    public static String toUpperCaseFirstOne(String s) {
        if (Character.isUpperCase(s.charAt(0))) {
            return s;
        } else {
            return (new StringBuilder())
                .append(Character.toUpperCase(s.charAt(0)))
                .append(s.substring(1)).toString();
        }
    }

    /**
     * replace:(替换字符串函数)
     *
     * @param strSource
     *      源字符串
     * @param strFrom
     *      要替换的子串
     * @param strTo
     *      替换为的字符串
     * @return
     * @since JDK 1.7
     */
    public static String replace(String strSource, String strFrom,
        String strTo) {
        // 如果要替换的子串为空, 则直接返回源串
        if (strFrom == null || strFrom.equals(""))
            return strSource;
        String strDest = "";
        // 要替换的子串长度
        int intFromLen = strFrom.length();
        int intPos;
        // 循环替换字符串
        while ((intPos = strSource.indexOf(strFrom)) != -1) {
            // 获取匹配字符串的左边子串
            strDest = strDest + strSource.substring(0, intPos);
            // 加上替换后的子串
            strDest = strDest + strTo;
            // 修改源串为匹配子串后的子串
        }
    }
}
```



```
        strSource = strSource.substring(intPos + intFromLen);
    }
    // 加上没有匹配的子串
    strDest = strDest + strSource;
    // 返回
    return strDest;
}

/**
 * String类型日期转为Date类型
 *
 * @param dateStr
 * @return
 * @throws ParseException
 * @throws Exception
 */
public static Date dateConvertFormat(String dateStr) throws ParseException {
    Date date = new Date();
    SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
    date = format.parse(dateStr);
    return date;
}
}
```

### Student.java

```
package com.wolfgy.domain;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

import org.hibernate.annotations.GenericGenerator;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@NoArgsConstructor
@Getter
@Setter
public class Student implements Serializable{

    /**
     * @fields serialVersionUID
     */

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(generator = "idGenerator")
    @GenericGenerator(name = "idGenerator", strategy = "uuid")
    private String id;
    private String sName;
}
```

### StudentService.java



```
//其余代码略，只贴出导入导出相关方法

@Override
public void exportExcel(String[] ids, OutputStream out) {

    String keyValue="学生ID:id,学生姓名:sName";
    List<Student> list = new ArrayList<>();
    for (int i = 0; i < ids.length; i++) {
        Student student = repository.findOne(ids[i]);

        list.add(student);
    }
    try {
        ExcelUtil.exportExcel("学生数据导出", out, ExcelUtil.getMap(keyValue), list, "com.
        logger.info("导出成功!");
    } catch (Exception e) {
        e.printStackTrace();
        logger.info("导出失败!");
    }
}

@Override
public void importExcel(String fileName,InputStream in) {
    logger.info("导入"+fileName+"开始");
    String keyValue="学生ID:id,学生姓名:sName";
    List<Student> list = null;
    try {
        //readExcel参数列表
        //fileName Excel文件名
        //inputStream 输入流
        //map 表头和属性的Map集合,其中Map中Key为Excel列的名称, Value为反射类的属性
        //classPath 需要映射的model的路径
        //rowNumIndex 表头所在行数
        list = ExcelUtil.readExcel("student.xlsx",in,ExcelUtil.getMap(keyValue),
            "com.wolfgy.domain.Student",1);
        logger.info(" 导入成功!");
    } catch (Exception e) {
        logger.error("导入失败!");
    }
    for (Student student : list) {
        logger.info("-----");
        logger.info("name:"+student.getSName());
    }
    logger.info(" 导入结束!");
}
}
```

#### AssociationController.java

```
//其余代码略，只贴出导入导出相关方法

@RequestMapping(value = "/student/export", method = RequestMethod.GET)
public void exporMemberFormExcel(@RequestParam(value = "ids", defaultValue = "", require
    logger.info("-----导出列表到Excel-----");
    res.setContentType("application/msexcel;charset=UTF-8");
    res.addHeader("Content-Disposition", "attachment;filename=members.xls");
    OutputStream out = res.getOutputStream();
    studentService.exportExcel(ids, out);
}

@RequestMapping(value="/studen/ajaxUpload.do",method=RequestMethod.POST)
public void ajaxUploadExcel(@RequestParam("file") MultipartFile file) throws Exception{
    if(!file.isEmpty()){
        InputStream in = file.getInputStream();
        String fileName = file.getName();
        studentService.importExcel(fileName,in);
    }else{
        throw new BizException("上传失败，因为文件是空的");
    }
}

}
```



以上。  
希望我的文章对你能有所帮助。  
我不能保证文中所有说法的百分百正确，但我能保证它们都是我的理解和感悟以及拒绝复制黏贴。  
有什么意见、见解或疑惑，欢迎留言讨论。

Java后端开发 (/nb/15062676) 举报文章 © 著作权归作者所有



三汪 (/u/edee856014f6)

写了 6020 字，被 11 人关注，获得了 44 个喜欢 (/u/edee856014f6)

+ 关注

被误解是表达者的宿命。

如果觉得我的文章对您有用，请随意赞赏。您的支持将鼓励我继续创作！

赞赏支持

喜欢 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-like-button) 10



更多分享

(http://cwb.assets.jianshu.io/notes/images/1727266)





登录 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-comment-form) 发表评论


评论

智慧如你，不想发表一点想法 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-nocomments-text)咩~

被以下专题收入，发现更多相似内容

- 

程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

Java服务器端编程 (/c/b5654f2880e2?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

Spring ... (/c/f0cf6eae1754?utm\_source=desktop&utm\_medium=notes-included-collection)

推荐阅读 更多精彩内容 > (/)

**基于Srting Boot和Apache POI的Excel导出实现示例(完美解决deprecated...**

本文不建议阅读。作者已有更好的解决方案:戳这里 前言 最近做公司项目，用到了POI实现Excel导入导出的功能。整个功能做下来，发现很多大牛的文章都已经过时，会报类似The method setBorderBottom(short)...

三汪 (/u/edee856014f6?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

JPA实体关系映射：@ManyToMany多对多关系、@OneToMany@ManyTo...

为什么要有实体关系映射 答：简化编程操作。把冗余的操作交给底层框架来处理。例如，如果我要给一位新入学的学生添加一位新的老师。而这个老师又是新来的，在学生数据库与教师数据库中均不存在对应的数...

三汪 (/u/edee856014f6?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

正是因为不够聪明，所以把这些提高效率的工作习惯坚...

(/p/a475c001115f?

正是因为不够聪明，所以把这些提高效率的工作习惯坚持了五年，然后，我升职了。对于我这样一个处世不够圆滑，说话不够好听，也不喜欢须溜拍马，甚...

utm\_campaign=maleskine&utm\_content=note&utm

韩老白 (/u/1fde17d765f5?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

想你，我只字不提 (/p/50b72c390405?utm\_campaign...

(/p/50b72c390405?

我望向钟楼，朝霞和寒露，人潮和车流 我在你的城市，在没有你的孤独里 我在你的身侧，在失去你的悲哀里 风吹起，紫色的清晨，泪珠晶莹，化为灰烬 你...

utm\_campaign=maleskine&utm\_content=note&utm

寒烟衰草 (/u/6f05d462e90c?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

曾帮我打架的兄弟，现在和我不再联系 (/p/53c78f2d50...

(/p/53c78f2d5016?

强哥是我最铁的兄弟，现在在德州开了几家扒鸡店。前段时间，强哥给我打电话说：“老三，我下周四结婚，你得来当伴郎。”那段时间我正处于低谷期。稿...

utm\_campaign=maleskine&utm\_content=note&utm

吕白Alex (/u/4b6bdf4f9b22?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

