# AutoPOI (Excel和 Word简易工具类 EasyPOI衍变升级版)

AutoPOI 功能如同名字auto，追求的就是自动化，让一个没接触过poi的人员，可以傻瓜式半智能化的快速实现Excel导入导出、Word模板导出、可以仅仅5行代码就可以完成Excel的导入导出。

## AutoPOI的主要特点

```
1.设计精巧,使用简单
2.接口丰富,扩展简单
3.默认值多,write less do more
4.AbstractView 支持,web导出可以简单明了
```

## AutoPOI的几个入口工具类

```
1.ExcelExportUtil Excel导出(普通导出,模板导出)
2.ExcelImportUtil Excel导入
3.WordExportUtil  Word导出(只支持docx ,doc版本poi存在图片的bug,暂不支持)
```

## 关于Excel导出XLS和XLSX区别

```
1.导出时间XLS比XLSX快2-3倍
2.导出大小XLS是XLSX的2-3倍或者更多
3.导出需要综合网速和本地速度做考虑^~^
```

## 几个工程的说明

```
1.autopoi-parent 父包--作用大家都懂得
2.autopoi 导入导出的工具包,可以完成Excel导出,导入,Word的导出,Excel的导出功能
3.autopoi-web  耦合了spring-mvc 基于AbstractView,极大的简化spring-mvc下的导出功能
4.sax 导入使用xercesImpl这个包(这个包可能造成奇怪的问题哈),word导出使用poi-scratchpad,都作为可选包了
```

## maven

```
    <dependency>
     <groupId>org.jeecgframework</groupId>
     <artifactId>autopoi-web</artifactId>
     <version>1.0.4</version>
    </dependency>
```

## AutoPoi 模板 表达式支持

- 空格分割
- 三目运算 {{test ? obj:obj2}}
- n: 表示 这个cell是数值类型 {{n:}}
- le: 代表长度{{le:()}} 在if/else 运用{{le:() > 8 ? obj1 : obj2}}
- fd: 格式化时间 {{fd:(obj;yyyy-MM-dd)}}
- fn: 格式化数字 {{fn:(obj;###.00)}}
- fe: 遍历数据,创建row
- !fe: 遍历数据不创建row
- $fe: 下移插入,把当前行,下面的行全部下移.size()行,然后插入
- !if: 删除当前列 {{!if:(test)}}
- 单引号表示常量值 '' 比如'1' 那么输出的就是 1

# AutoPoi导出实例

1.注解,导入导出都是基于注解的,实体上做上注解,标示导出对象,同时可以做一些操作

```
    @ExcelTarget("courseEntity")
    public class CourseEntity implements java.io.Serializable {
    /** 主键 */
    private String id;
    /** 课程名称 */
    @Excel(name = "课程名称", orderNum = "1", needMerge = true)
    private String name;
    /** 老师主键 */
    @ExcelEntity(id = "yuwen")
    @ExcelVerify()
    private TeacherEntity teacher;
    /** 老师主键 */
    @ExcelEntity(id = "shuxue")
    private TeacherEntity shuxueteacher;

    @ExcelCollection(name = "选课学生", orderNum = "4")
    private List<StudentEntity> students;
```

2.基础导出

传入导出参数,导出对象,以及对象列表即可完成导出

```
    HSSFWorkbook workbook = ExcelExportUtil.exportExcel(new ExportParams(
                "2412312", "测试", "测试"), CourseEntity.class, list);
```

```
    HSSFWorkbook workbook = ExcelExportUtil.exportExcel(new ExportParams(
                "2412312", "测试", "测试"), CourseEntity.class, list,exportFields);
```

3.基础导出,带有索引

在到处参数设置一个值,就可以在导出列增加索引

```
    ExportParams params = new ExportParams("2412312", "测试", "测试");
    params.setAddIndex(true);
    HSSFWorkbook workbook = ExcelExportUtil.exportExcel(params,
            TeacherEntity.class, telist);
```

4.导出Map

创建类似注解的集合,即可完成Map的导出,略有麻烦

```
    List<ExcelExportEntity> entity = new ArrayList<ExcelExportEntity>();
    entity.add(new ExcelExportEntity("姓名", "name"));
    entity.add(new ExcelExportEntity("性别", "sex"));

    List<Map<String, String>> list = new ArrayList<Map<String, String>>();
    Map<String, String> map;
    for (int i = 0; i < 10; i++) {
        map = new HashMap<String, String>();
        map.put("name", "1" + i);
        map.put("sex", "2" + i);
        list.add(map);
    }

    HSSFWorkbook workbook = ExcelExportUtil.exportExcel(new ExportParams(
            "测试", "测试"), entity, list);
```

5.模板导出

根据模板配置,完成对应导出

```
TemplateExportParams params = new TemplateExportParams();
params.setHeadingRows(2);
params.setHeadingStartRow(2);
Map<String,Object> map = new HashMap<String, Object>();
map.put("year", "2013");
map.put("sunCourses", list.size());
Map<String,Object> obj = new HashMap<String, Object>();
map.put("obj", obj);
obj.put("name", list.size());
 params.setTemplateUrl("org/jeecgframework/poi/excel/doc/exportTemp.xls");
Workbook book = ExcelExportUtil.exportExcel(params, CourseEntity.class, list,
        map);
```

6.导入

设置导入参数,传入文件或者流,即可获得相应的list

```
ImportParams params = new ImportParams();
params.setTitleRows(2);
params.setHeadRows(2);
//params.setSheetNum(9);
params.setNeedSave(true);
long start = new Date().getTime();
List<CourseEntity> list = ExcelImportUtil.importExcel(new File(
        "d:/tt.xls"), CourseEntity.class, params);
```

7.SpringMvc的无缝融合

简单几句话,Excel导出搞定

```
@RequestMapping(value = "/exportXls")
public ModelAndView exportXls(HttpServletRequest request, HttpServletResponse response) {
    ModelAndView mv = new ModelAndView(new JeecgEntityExcelView());
    List<JeecgDemo> pageList = jeecgDemoService.list();
    //导出文件名称
    mv.addObject(NormalExcelConstants.FILE_NAME,"导出Excel文件名字");
    //注解对象Class
    mv.addObject(NormalExcelConstants.CLASS,JeecgDemo.class);
    //自定义导出字段         mv.addObject(NormalExcelConstants.EXPORT_FIELDS,"name,keyWord,punchTime
    //自定义表格参数
    mv.addObject(NormalExcelConstants.PARAMS,new ExportParams("自定义导出Excel模板内容标题", "自定义She
    //导出数据列表
    mv.addObject(NormalExcelConstants.DATA_LIST,pageList);
    return mv;
}
```

| 自定义视图 | 用途 | 描述 |
| --- | --- | --- |
| JeecgMapExcelView | 实体对象导出视图 | 例如：List |
| JeecgEntityExcelView | Map对象导出视图 | List<Map<String, String>> list |
| JeecgTemplateExcelView | Excel模板导出视图 | - |
| JeecgTemplateWordView | Word模板导出视图 | - |

8.Excel导入校验,过滤不符合规则的数据,追加错误信息到Excel,提供常用的校验规则,已经通用的校验接口

```
/**
 * Email校验
 */
@Excel(name = "Email", width = 25)
@ExcelVerify(isEmail = true, notNull = true)
private String email;
/**
 * 手机号校验
 */
@Excel(name = "Mobile", width = 20)
```

```
    @ExcelVerify(isMobile = true, notNull = true)
    private String mobile;

    ExcelImportResult<ExcelVerifyEntity> result = ExcelImportUtil.importExcelVerify(new File(
            "d:/tt.xls"), ExcelVerifyEntity.class, params);
    for (int i = 0; i < result.getList().size(); i++) {
        System.out.println(ReflectionToStringBuilder.toString(result.getList().get(i)));
    }
```

9.导入Map

设置导入参数,传入文件或者流,即可获得相应的list,自定义Key,需要实现IExcelDataHandler接口

```
    ImportParams params = new ImportParams();
    List<Map<String,Object>> list = ExcelImportUtil.importExcel(new File(
            "d:/tt.xls"), Map.class, params);
```

10.字典用法

在实体属性注解excel中添加dicCode="",此处dicCode即为jeecg系统中数据字典的Code

```
   @Excel(name="性别",width=15,dicCode="sex")
   private java.lang.String sex;
```

11.字典表用法

此处dictTable为数据库表名，dicCode为关联字段名，dicText为excel中显示的内容对应的字段

```
    @Excel(name="部门",dictTable="t_s_depart",dicCode="id",dicText="departname")
    private java.lang.String depart;
```

12.Replace用法

若数据库中存储的是0/1 ， 则导出/导入的excel单元格中显示的是女/男

```
    @Excel(name="测试替换",width=15,replace={"男_1","女_0"})
    private java.lang.String fdReplace;
```

13.高级字段转换用法

- exportConvert：在导出的时候需要替换值则配置该值为true，同时增加一个方法，方法名为原get方法名前加convert。
- importConvert：在导入的时候需要替换值则配置该值为true，同时增加一个方法，方法名为原set方法名前加convert。

```
    @Excel(name="测试转换",width=15,exportConvert=true,importConvert=true)
    private java.lang.String fdConvert;

    /**
      * 转换值示例：  在该字段值的后面加上元
      * @return
      */
    public String convertgetFdConvert(){
      return this.fdConvert+"元";
    }

    /**
     * 转换值示例：  替换掉excel单元格中的"元"
     * @return
     */
    public void convertsetFdConvert(String fdConvert){
      this.fdConvert = fdConvert.replace("元","");
    }
```

## Excel 注解说明

@Excel

| 属性 | 类型 | 默认值 | 功能 |
| --- | --- | --- | --- |
| name | String | null | 列名,支持name_id |
| needMerge | boolean | fasle | 是否需要纵向合并单元格(用于含有list中,单个的单元格,合并list创建的多个row) |
| orderNum | String | "0" | 列的排序,支持name_id |
| replace | String[] | {} | 值得替换 导出是{a_id,b_id} 导入反过来 |
| savePath | String | "upload" | 导入文件保存路径,如果是图片可以填写,默认是upload/className/ IconEntity 这个类对应的就是upload/Icon/ |
| type | int | 1 | 导出类型 1 是文本 2 是图片,3 是函数,10 是数字 默认是文本 |
| width | double | 10 | 列宽 |
| height | double | 10 | 列高,后期打算统一使用@ExcelTarget的height,这个会被废弃,注意 |
| isStatistics | boolean | fasle | 自动统计数据,在追加一行统计,把所有数据都和输出 这个处理会吞没异常,请注意这一点 |
| isHyperlink | boolean | FALSE | 超链接,如果是需要实现接口返回对象 |
| isImportField | boolean | TRUE | (作废参数)校验字段,看看这个字段是不是导入的Excel中有,如果没有说明是错误的Excel,读取失败,支持name_id |
| exportFormat | String | "" | 导出的时间格式,以这个是否为空来判断是否需要格式化日期 |
| importFormat | String | "" | 导入的时间格式,以这个是否为空来判断是否需要格式化日期 |
| format | String | "" | 时间格式,相当于同时设置了exportFormat 和 importFormat |
| databaseFormat | String | "yyyyMMddHHmmss" | 导出时间设置,如果字段是Date类型则不需要设置 数据库如果是string 类型,这个需要设置这个数据库格式,用以转换时间格式输出 |
| numFormat | String | "" | 数字格式化,参数是Pattern,使用的对象是DecimalFormat |
| imageType | int | 1 | 导出类型 1 从file读取 2 是从数据库中读取 默认是文件 同样导入也是一样的 |
| suffix | String | "" | 文字后缀,如% 90 变成90% |
| isWrap | boolean | TRUE | 是否换行 即支持\n |
| mergeRely | int[] | {} | 合并单元格依赖关系,比如第二列合并是基于第一列 则{0}就可以了 |

| 属性 | 类型 | 默认值 | 功能 |
| --- | --- | --- | --- |
| mergeVertical | boolean | fasle | 纵向合并内容相同的单元格 |
| fixedIndex | int | -1 | 对应excel的列,忽略名字 |
| isColumnHidden | boolean | FALSE | 导出隐藏列 |

@ExcelCollection

| 属性 | 类型 | 默认值 | 功能 |
| --- | --- | --- | --- |
| id | String | null | 定义ID |
| name | String | null | 定义集合列名,支持nanm_id |
| orderNum | int | 0 | 排序,支持name_id |
| type | Class<?> | ArrayList.class | 导入时创建对象使用 |

单表导出实体注解源码

```
public class SysUser implements Serializable {

    /**id*/
    private String id;

    /**登录账号 */
    @Excel(name = "登录账号", width = 15)
    private String username;

    /**真实姓名*/
    @Excel(name = "真实姓名", width = 15)
    private String realname;

    /**头像*/
    @Excel(name = "头像", width = 15)
    private String avatar;

    /**生日*/
    @Excel(name = "生日", width = 15, format = "yyyy-MM-dd")
    private Date birthday;

    /**性别（1：男 2：女）*/
    @Excel(name = "性别", width = 15,dicCode="sex")
    private Integer sex;

    /**电子邮件*/
    @Excel(name = "电子邮件", width = 15)
    private String email;

    /**电话*/
    @Excel(name = "电话", width = 15)
    private String phone;

    /**状态(1：正常  2：冻结 ）*/
    @Excel(name = "状态", width = 15,replace={"正常_1","冻结_0"})
    private Integer status;
```

一对多导出实体注解源码

```
@Data
public class JeecgOrderMainPage {

    /**主键*/
    private java.lang.String id;
    /**订单号*/
    @Excel(name="订单号",width=15)
    private java.lang.String orderCode;
    /**订单类型*/
```

```
    private java.lang.String ctype;
    /**订单日期*/
    @Excel(name="订单日期",width=15,format = "yyyy-MM-dd")
    private java.util.Date orderDate;
    /**订单金额*/
    @Excel(name="订单金额",width=15)
    private java.lang.Double orderMoney;
    /**订单备注*/
    private java.lang.String content;
    /**创建人*/
    private java.lang.String createBy;
    /**创建时间*/
    private java.util.Date createTime;
    /**修改人*/
    private java.lang.String updateBy;
    /**修改时间*/
    private java.util.Date updateTime;

    @ExcelCollection(name="客户")
    private List<JeecgOrderCustomer> jeecgOrderCustomerList;
    @ExcelCollection(name="机票")
    private List<JeecgOrderTicket> jeecgOrderTicketList;
}
```

上一篇：针对敏感数据，加密传递方案

下一篇：redis 如何使用?