# 简要教程

目录

[简要教程 1](#_Toc441066811)

[前面的话 1](#_Toc441066812)

[第一章 Excel篇 2](#_Toc441066813)

[Excel的注解 2](#_Toc441066814)

[Excel导出 6](#_Toc441066815)

[Excel模板导出 11](#_Toc441066816)

[Excel导入 14](#_Toc441066817)

[第二章 Word篇 16](#_Toc441066818)

[Word模板导出 16](#_Toc441066819)

[第三章PDF导出 17](#_Toc441066820)

[第四章HTML导出 18](#_Toc441066821)

[HTML导出 18](#_Toc441066822)

[第五章 Excel 图表 19](#_Toc441066823)

[第六章 缓存以及工具类 19](#_Toc441066824)

修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 日期 | 版本 | 作者 | 修订类型 | 描述 |
| 2016-1-20 | 1．0 | JueYue | 新建 |  |
| 2016-1-22 | 1.1 | JueYue | 修改 | 导入校验,文件合法性校验 |

## 前面的话

EasyPoi的编写其实是一次意外,之前我不太愿意写导入导出,因为代码号复杂,每次一个Excel 都要写几百行,仅有少量的复用,一次需要写许多的导入导出,又没有人手,正好看到了Jeecg对应Poi的一个封装,但是他的封装比较简单,就自己在他思想的基础上开始构建现在的项目了.

EasyPoi秉承思想就是尽量简单,入库少,可以很快把简单的工作干了这是第一个版本的功能,后来慢慢用的人多了,也就功能慢慢丰富了,现在包含了Excel的导入导出,Word的导出,PDF的导出,Excel Charts的导出, Html的导出5个功能模块,重点还是Excel,毕竟Excel是最常用的.

EasyPoi已经走过了3年,未来也会继续走,保持现有简单的功能继续前进.

程序员你懂得,懒得测试,目前测试覆盖率,基本功能我都写了,没有的就遇到了再说吧



## Excel篇

### Excel的注解

本来注解是整个模块的精髓,不过随着模板的使用,他的重要性也就不如以前了,不过还是最简单的使用方式.注解都是在easypoi-annotation这个jar,拆分开的目的就是为了让多模块的maven项目少引用,因为easypoi-annotation是没有任何依赖的,只是单纯的注解

#### @Excel Excel 导出基本注释

这个注解是整个的基石,简单项目只靠这一个注解就可以完成所有功能

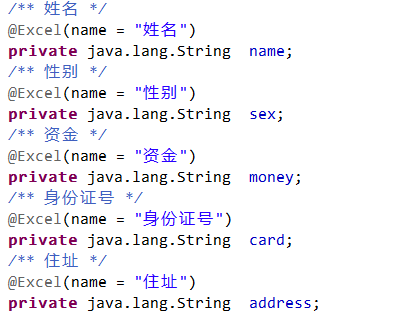
|  |  |
| --- | --- |
| **限定符和类型** | **可选元素和说明** |
| java.lang.String | [**databaseFormat**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#databaseFormat--)  导出时间设置,如果字段是Date类型则不需要设置 数据库如果是string 类型,这个需要设置这个数据库格式 |
| java.lang.String | [**exportFormat**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#exportFormat--)  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 |
| java.lang.String | [**format**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#format--)  时间格式,相当于同时设置了exportFormat 和 importFormat |
| double | [**height**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#height--)  导出时在excel中每个列的高度 单位为字符，一个汉字=2个字符 |
| int | [**imageType**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#imageType--)  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 |
| java.lang.String | [**importFormat**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#importFormat--)  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 |
| boolean | [**isHyperlink**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#isHyperlink--)  这个是不是超链接,如果是需要实现接口返回对象 |
| java.lang.String | [**isImportField**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#isImportField--)  导入时会校验这个字段,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 本意是想用true的,想想还是false比较好 可以使用a\_id,b\_id来确实是否使用 |
| boolean | [**isStatistics**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#isStatistics--)  是否自动统计数据,如果是统计,true的话在最后追加一行统计,把所有数据都和 这个处理会吞没异常,请注意这一点 |
| boolean | [**isWrap**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#isWrap--)  是否换行 即支持\n |
| int[] | [**mergeRely**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#mergeRely--)  合并单元格依赖关系,比如第二列合并是基于第一列 则{1}就可以了 |
| boolean | [**mergeVertical**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#mergeVertical--)  纵向合并内容相同的单元格 |
| boolean | [**needMerge**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#needMerge--)  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) |
| java.lang.String | [**orderNum**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#orderNum--)  展示到第几个可以使用a\_id,b\_id来确定不同排序 |
| java.lang.String[] | [**replace**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#replace--)  值得替换 导出是{a\_id,b\_id} 导入反过来,所以只用写一个 |
| java.lang.String | [**savePath**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#savePath--)  导入路径,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/ |
| java.lang.String | [**suffix**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#suffix--)  文字后缀,如% 90 变成90% |
| int | [**type**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#type--)  导出类型 1 是文本 2 是图片,3是函数 默认是文本 |
| double | [**width**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#width--)  导出时在excel中每个列的宽 单位为字符，一个汉字=2个字符 如 以列名列内容中较合适的长度 例如姓名列6 【姓名一般三个字】 性别列4【男女占1，但是列标题两个汉字】 限制1-255 |

这当中需要说明的是:

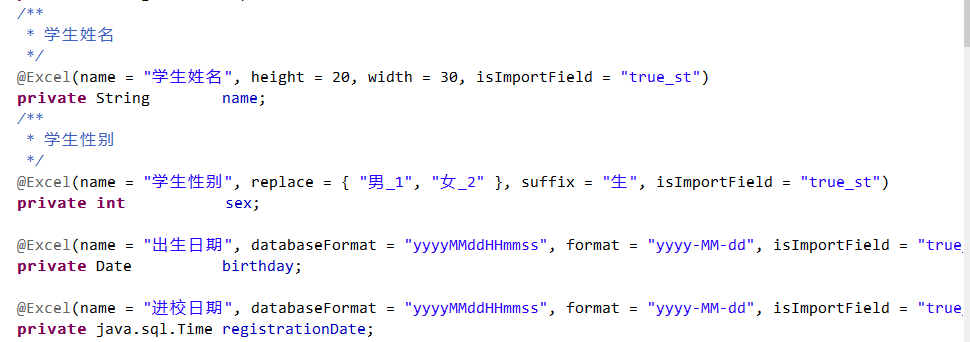
[mergeVertical](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#mergeVertical--)和[mergeRely](file:///D:\\mySpace\\myself\\easypoi\\easypoi-annotation\\target\\apidocs\\org\\jeecgframework\\poi\\excel\\annotation\\Excel.html" \l "mergeRely--)这两者是一起存在的功能,就是纵向合并单元和, [mergeRely](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\Excel.html#mergeRely--)是判断和前面单元格的依赖,可用于多种集合,不太好导出情况,来合并处理

DEMO:

最简单的导出



复杂点的导出:

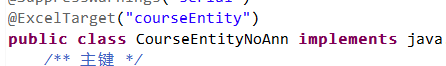


#### @ExcelTarget 标记导入ID

就一个功能标记ID,用以区别多个导出对象,便于一个对象用于多个导出实例

|  |  |
| --- | --- |
| java.lang.String | [**value**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\ExcelTarget.html#value--)  定义excel导出ID 来限定导出字段 |

DEMO



使用方法



#### @ExcelEntity 用来标示导出对象

因为对象不是基础属性,需要继续判断,所以加入这个属性,这个属性可以无限迭代,

ID和ExcelTarget的ID功能一致

|  |  |
| --- | --- |
| java.lang.String | [**id**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\ExcelEntity.html#id--)  定义excel导出ID 来限定导出字段,处理一个类对应多个不同名称的情况 |
| java.lang.String | [**name**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\ExcelEntity.html#name--)  导出时，对应数据库的字段 主要是用户区分每个字段， 不能有annocation重名的 导出时的列名 导出排序跟定义了annotation的字段的顺序有关 可以使用a\_id,b\_id来确实是否使用 |

#### @ExcelCollection 标示集合

集合只能一层,如果name有值,表头就会有两行,有过没值就会只有一行

|  |  |
| --- | --- |
| java.lang.String | [**id**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\ExcelCollection.html#id--)  定义excel导出ID 来限定导出字段,处理一个类对应多个不同名称的情况 |
| java.lang.String | [**orderNum**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\ExcelCollection.html#orderNum--)  展示到第几个同样可以使用a\_id,b\_id |
| java.lang.Class<?> | [**type**](file:///D:\mySpace\myself\easypoi\easypoi-annotation\target\apidocs\org\jeecgframework\poi\excel\annotation\ExcelCollection.html#type--)  创建时创建的类型 默认值是 arrayList |

DEMO









#### @ExcelIgnore 忽略字段

标记为excel 创建实体忽略,放置死循环的造成

注解基本上就到这来完了,主要还是各个注解直接的搭配使用,核心还是在于@Excel,主要的功能也是在这个地方,请大家开发之前先看看这个吧

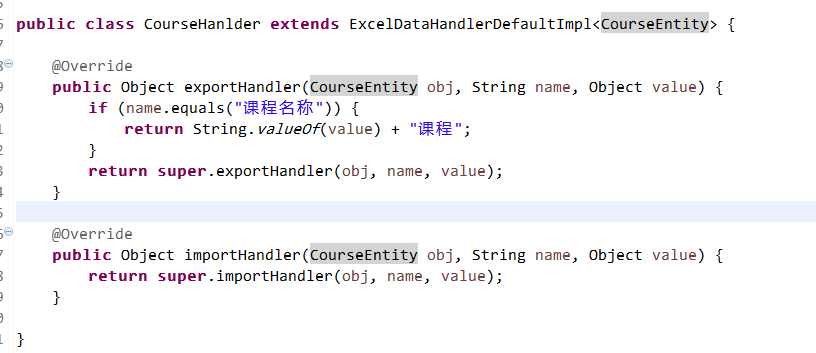
### Excel导出

导出统一入口 ExcelExportUtil

**excel,导入导出,模板导出的基础参数,数据处理的接口**

|  |  |
| --- | --- |
| private **[IExcelDataHandler](file:///D:\\mySpace\\myself\\easypoi\\easypoi-base\\doc\\org\\jeecgframework\\poi\\handler\\inter\\IExcelDataHandler.html" \o "org.jeecgframework.poi.handler.inter中的接口)** | [**dataHanlder**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExcelBaseParams.html#dataHanlder)  数据处理接口,以此为主,replace,format都在这后面 |

默认实现ExcelDataHandlerDefaultImpl ,用户自定义的实现可以继承这个



当然你可以可以用spring来管理这个对象,都是一样的

#### ExportParams 导出参数

注解导出和map导出的参数设置

|  |  |
| --- | --- |
| private boolean | [**addIndex**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#addIndex)  是否添加序列 |
| private short | [**color**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#color)  表头颜色 |
| private java.lang.String[] | [**exclusions**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#exclusions)  过滤的属性 |
| private int | [**freezeCol**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#freezeCol)  冰冻列 |
| private short | [**headerColor**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#headerColor)  属性说明行的颜色 例如:HSSFColor.SKY\_BLUE.index 默认 |
| private java.lang.String | [**indexName**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#indexName)  是否添加需要需要 |
| private boolean | [**isAppendGraph**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#isAppendGraph)  是否追加图形 |
| private boolean | [**isCreateHeadRows**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#isCreateHeadRows)  是否创建表头 |
| private boolean | [**isDynamicData**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#isDynamicData)  是否动态获取数据 |
| private java.lang.String | [**secondTitle**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#secondTitle)  第二行名称 |
| private short | [**secondTitleHeight**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#secondTitleHeight)  表格名称 |
| private java.lang.String | [**sheetName**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#sheetName)  sheetName |
| private java.lang.Class<?> | [**style**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#style)  Excel 导出style |
| private java.lang.String | [**title**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#title)  表格名称 |
| private short | [**titleHeight**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#titleHeight)  表格名称 |
| private **[ExcelType](file:///D:\\mySpace\\myself\\easypoi\\easypoi-base\\doc\\org\\jeecgframework\\poi\\excel\\entity\\enmus\\ExcelType.html" \o "org.jeecgframework.poi.excel.entity.enmus中的枚举)** | [**type**](file:///D:\mySpace\myself\easypoi\easypoi-base\doc\org\jeecgframework\poi\excel\entity\ExportParams.html#type)  Excel 导出版本 |

#### 对象注解导出

注解导出 第一步先给对象加上注解,这个我就复述了,之后查出来对象列表,这个大家也都懂



就一个方法,大数据量推荐使用2007版本,效率高,生成文件小,

Class 填的就是Collection中的对象

DEMO



#### Map的导出

有时候我们希望导出一个map集合或者我们导出的列表是不固定的,就需要这个导出了



和上面方法的区别就是上面的Classes变成了EntityList 其实Classes最好也会翻译成EntityList,只是自动帮你翻译了,这个方法就是把控制器还给你了,但是这个就比较麻烦,提供4个构造器name就是列的名称,key对应就是map中的key或者属性的name,便于map的快速导出

这个也是基本上支持注解中的所有功能

|  |  |
| --- | --- |
| [**ExcelExportEntity**](file:///D:\mySpace\myself\easypoi\easypoi-base\target\apidocs\org\jeecgframework\poi\excel\entity\params\ExcelExportEntity.html#ExcelExportEntity--)() |  |
| [**ExcelExportEntity**](file:///D:\mySpace\myself\easypoi\easypoi-base\target\apidocs\org\jeecgframework\poi\excel\entity\params\ExcelExportEntity.html#ExcelExportEntity-java.lang.String-)(java.lang.String name) |  |
| [**ExcelExportEntity**](file:///D:\mySpace\myself\easypoi\easypoi-base\target\apidocs\org\jeecgframework\poi\excel\entity\params\ExcelExportEntity.html#ExcelExportEntity-java.lang.String-java.lang.Object-)(java.lang.String name, java.lang.Object key) |  |
| [**ExcelExportEntity**](file:///D:\mySpace\myself\easypoi\easypoi-base\target\apidocs\org\jeecgframework\poi\excel\entity\params\ExcelExportEntity.html#ExcelExportEntity-java.lang.String-java.lang.Object-int-)(java.lang.String name, java.lang.Object key, int width) |  |

DEMO



#### 一次导出多个对象

其实就是对第一个方法的一个封装,没有什么需要介绍的



#### 定义样式

自定义Cell样式接口,方便大家自己处理Excel的样式



提供了三个实现,

ExcelExportStylerBorderImpl 有边框样式

ExcelExportStylerColorImpl 有边框间隔行样式

ExcelExportStylerDefaultImpl 默认的样式

样式使用方法ExportParams 中调用setStyler 就可以了,必须实现有参构造器

**public** ExcelExportStylerBorderImpl(Workbook workbook) {

**super**.createStyles(workbook);

}

### Excel模板导出

为了更好的导出Excel,以及Excel的样式定义,复杂的表头等功能对已代码设置的很复杂的情况下使用模板确实是更好的办法,下面这样的样式我们自己实现要实现好久,但是模板就比较容易实现了,Excel分分钟的事情,模板秉承的是继承你的Cell样式,尽量不破坏你的东西



#### 模板支持的语法

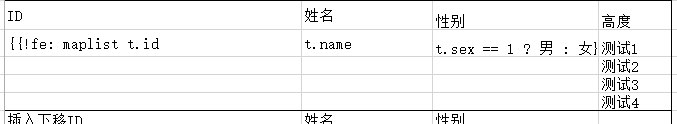
1. 空格分割
2. 三目运算 {{test ? obj:obj2}}
3. n: 表示 这个cell是数值类型 {{n:}}
4. le: 代表长度{{le:()}} 在if/else 运用{{le:() > 8 ? obj1 : obj2}}
5. fd: 格式化时间 {{fd:(obj;yyyy-MM-dd)}}
6. fn: 格式化数字 {{fn:(obj;###.00)}}
7. 单引号表示常量值 '' 比如'1' 那么输出的就是 1
8. fe: 遍历数据,创建row
9. !fe: 遍历数据不创建row
10. $fe: 下移插入,把当前行,下面的行全部下移.size()行,然后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

语法就上面这些,前面7个都没啥需要讲的都是我们一样的语法,说下下面几个

8,9,10都是一样的就是for循环创建表格,不同的是fe是全部都创建自己的表格,!fe是先使用现有的表格,不够了再创建表格,$fe是把这样之下的先往下移动X行,在便利,说下3个的场景

第一个不说了,是不是都要创建

第二个是:比如我们就创建4行或者<4行,4行之后还有别的数据,那么我们就使用!fe来迭代



生成后不会影响其他数据



第三个是:当遍历行下面仍有数据时,不影响下面的数据,进行插入操作

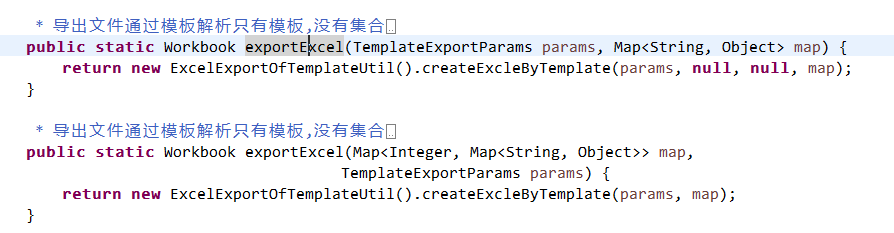


输出结果



#### 导出操作

提供两个方法,一个是都是模板,第二个是第一个封装,就是多个sheet的模板导出

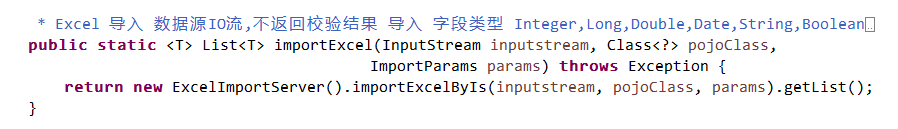


我的使用方式,如果是样式复杂就是模板,样式简单就是代码导出的

### Excel导入

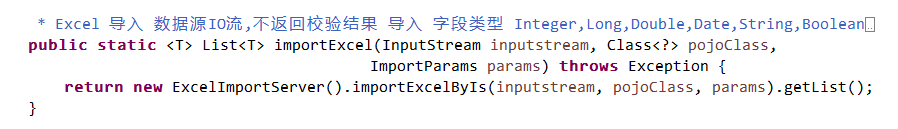
#### 注解导入

基于Excel注解导出,只要反过来看代码就可以了,基本上不用讲什么,导出支持的功能,基本上导入同样支持



#### Map导入

传入对象传入map就可以,返回Map<String,Object> 这个Object请大家注意了不确定是什么不过都是基本类型



和上面的是同一个方法

#### Sax导入

支持的功能不如上面的丰富,不在会吃图片,紧支持基本类型,

对于大数据量导入请使用Sax,避免造成内存溢出

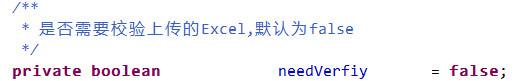


#### Excel数据校验

校验是集成的JSR303,可选hibernate或者Apache的实现,这个我就不讲了,请大家自行百度,

功能都是一样的,需要开启设置在ImportParams中

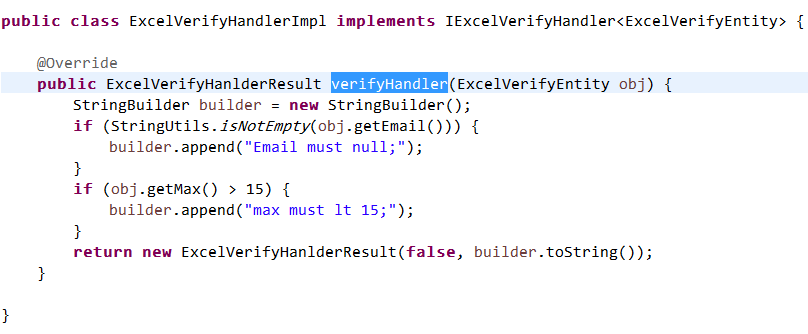
这是一个事前校验,一般是数据的合法性,针对数据唯一性什么的,可以使用接口校验



同时提供了导入校验的接口,是整个对象一起校验的,返回校验结果就可以,比较简单

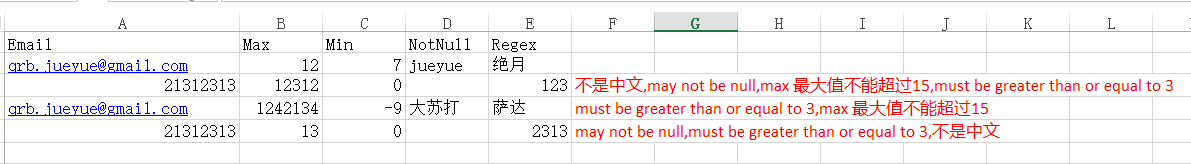


比如



下面说下错误信息的处理

错误信息默认会追加到这一行最后创建一个Cell中去,就是你可以返回这个Excel给用户,再让他改,

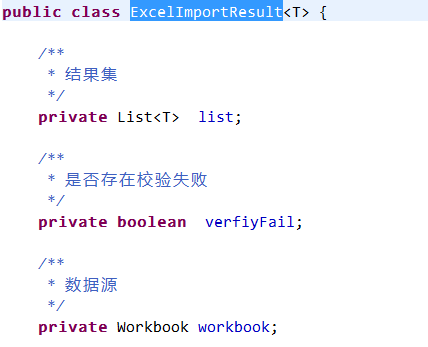


同时,我们也希望存放到数据库中这是我们需要我们的Entity实现一个接口



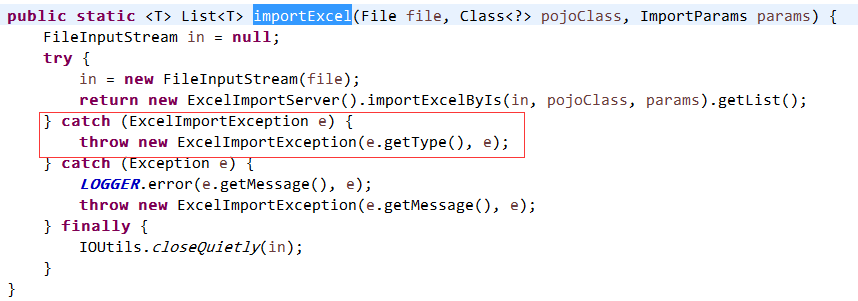
这样我们会把错误信息追加到你的对象中用于保存到数据库中

同时,会返回校验Result对象,用于判断是不是有错误,集合,已经Excel



#### 文件合法性校验

文件合法性校验,是校验我们的文件是不是我们给对方的模板,是否包含全了我们要求的列



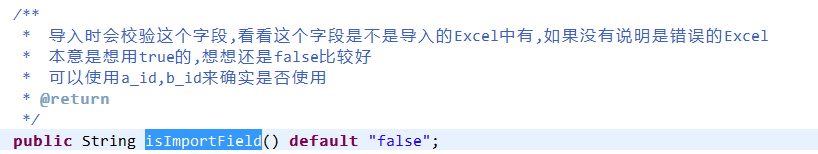
这个校验是中断校验,发现校验失败后会爆出异常,异常枚举值



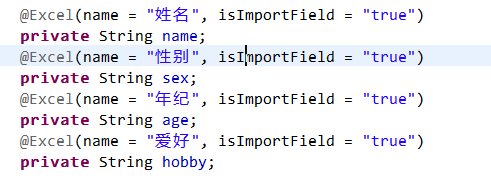
校验方法有两种,1.注解校验,2数组校验,推荐后者

1. 注解校验

在注解中加入



DEMO:



1. 数据值校验,

就是在导入参数中动态设置需要校验的对象比如:



如果是有集合也可以校验



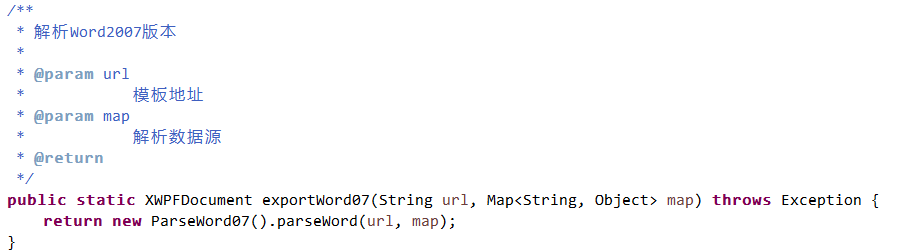
同样抛出校验异常信息,终止导入流程

## Word篇

### Word模板导出

因为POI的Word 2003支持的很不好,我也就不进行支持了,现在只支持2007版本

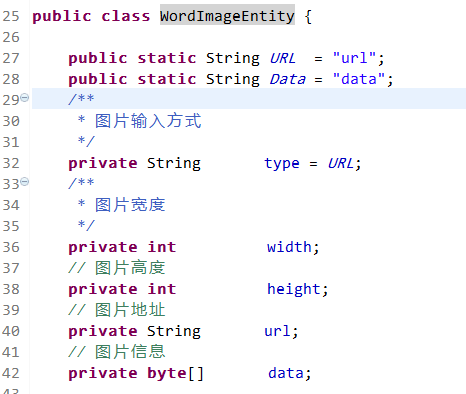
支持值替换,图片替换,table等功能



基本功能是基于Excel 模板导出功能的clone,实现了Excel模板中的大部分功能,语法也都是保持一致,目前不能嵌套table请注意

语法啥的就不讲了,说下不同的地方

提供一个专用的图片替换类,判断这个对象会把这个地方替换成图片

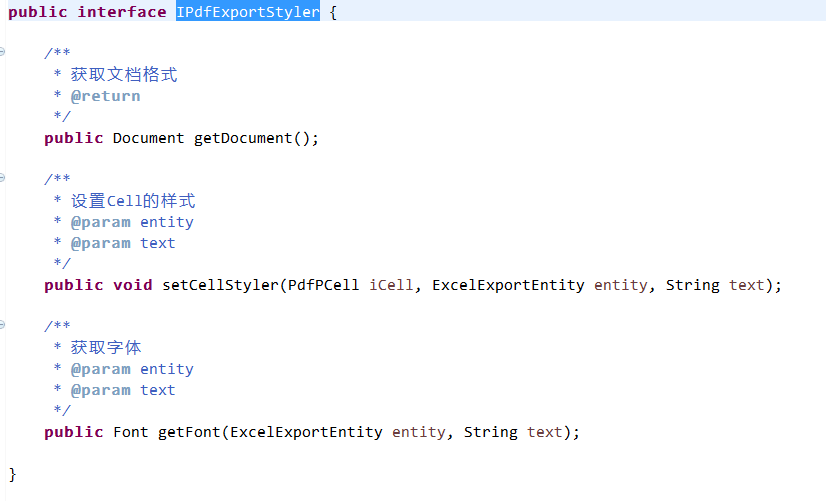


## 第三章PDF导出

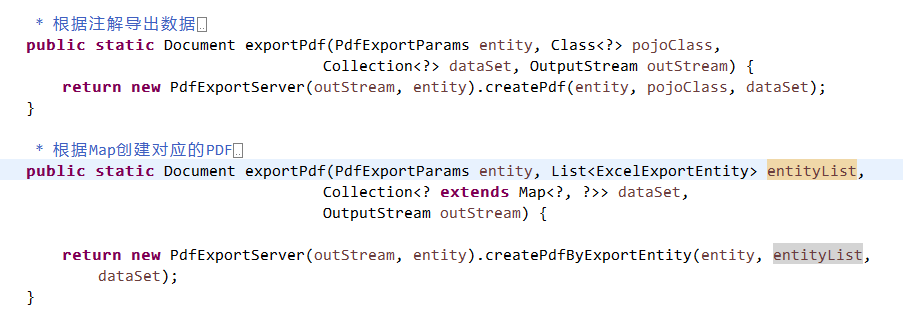
#### Pdf导出

同样是基于Excel模块的功能扩展,目前只支持Excel注解导出和map导出,还没支持模板

提供了样式接口IPdfExportStyler,用于创建Cell时候的样式设置



方法也和Excel;类似,但是要提供流



## 第四章HTML导出

### HTML导出

目前只支持Excel转为HTML,效果还算凑合,给用户看看可以,企业级还是需要专业的解决方案

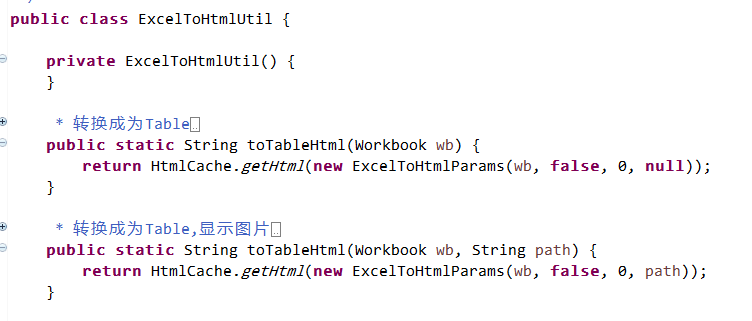


导出返回两种界面,支持图片转换

toTableHtml 只返回table对象

toAllHtml 返回包含html的对象

导出代码,算是一个单独的工具类



## 第五章 Excel 图表

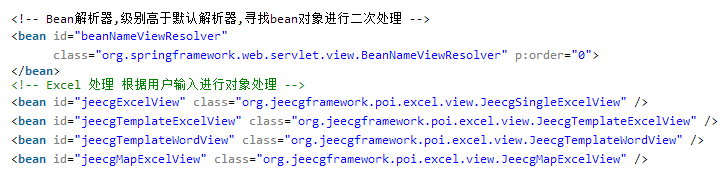
## 第六章 Spring web集成

### Spring mvc 的配置

使用spring mvc 需要在spring-mvc.xml或者其他的配置文件中进行配置,主要是

默认视图级别设置低点

Bean视图设置级别高一些,然后把我们的视图配置上,就完成了



2.0.8版本后加上了@Controller 里面只要在

*<*context:component-scan base-package="org.jeecgframework.poi.excel.view">

加入就可以了,但是解析器还是要配置的

这样就算集成完成了,下面我们介绍各种VIEW

### 注解view JeecgSingleExcelView

## 第七章 缓存以及工具类

#### 缓存

缓存是基于guava来实现的

提供了excel模板读取缓存,Html缓存,word模板读取缓存,图片读取缓存

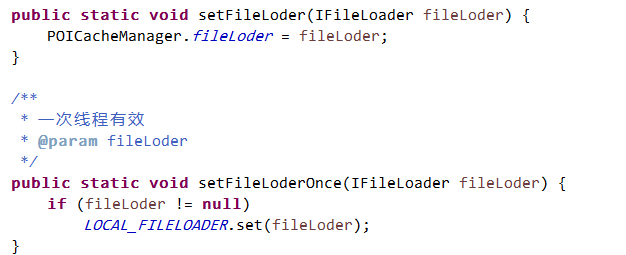
讲下自定义文件读取

IFileLoader 提供文件读取接口,自己实现这个接口,比如



然后根据自己的需要可以把自己实现的类设置为全局的类,或者线程内使用的类

POICacheManager提供对应的方法



|  |  |  |
| --- | --- | --- |
| 对象 | 缓存时间 | 最大缓存数量 |
| Excel,Word | 1小时 | 50 |
| Html | 7天 | 200个 |
| Image | 1天 | 200个 |

每个cache都提供了set方法,以便大家设置自己的实现

#### 工具类

PoiReflectorUtil 反射工具类,从mybatis上copy下来的

PoiMergeCellUtil 纵向合并工具类 合并内容相同的单元格

PoiCellUtil 读取单元格值,判断合并情况