# Easy-Poi学习文档

背景：工作中需要通过代码把数据插入到excel中，并且excel中展示的格式多种多样，

但是在代码中设置excel格式（表头、多行合并等）复杂又麻烦。为了使导出的excel跟需求给出的模板保持一致，采用easypoi的方式进行模板化。

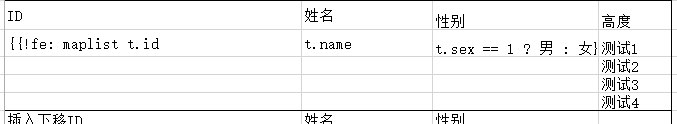
该方法需要在excel模板中指定函数名和语法，在代码中为该函数名写入数据即可。

#### 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:}}
   * 用来处理数字问题,防止上面有小叹号
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)}}
   * 使用java的时间格式化,进行格式化处理
6. fn: 格式化数字 {{fn:(obj;###.00)}}
   * 处理小数点问题
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. #fe: 行遍历
    * 其他都是列遍历,这个是横向遍历
12. !if: 删除当前列 {{!if:(test)}}
13. &NULL& 控制
14. ]] 换行符
    * 解决多行遍历的问题,详情看下嘛的遍历标签

#### 遍历标签的使用

1、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



2、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



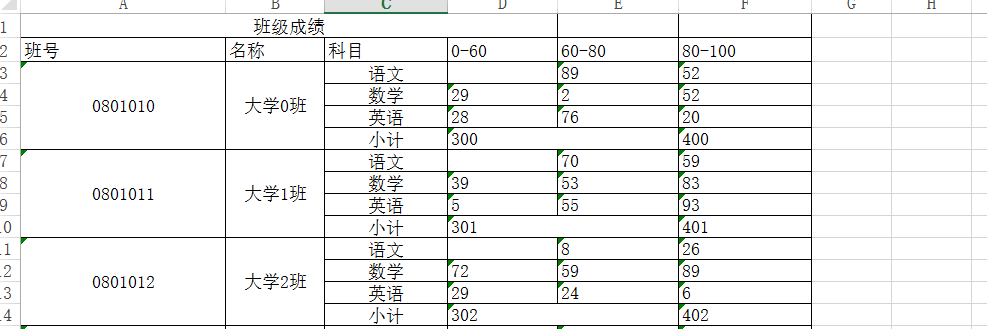
输出结果



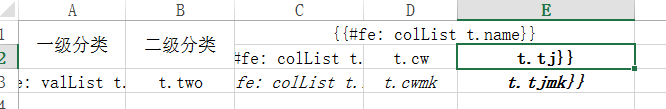
3、多行遍历

针对多行遍历问题,给出了]] 作为换行符用来标记list还没执行完的问题,多行遍历解决了部分Excel处理难的问题



生产效果

横向遍历



效果

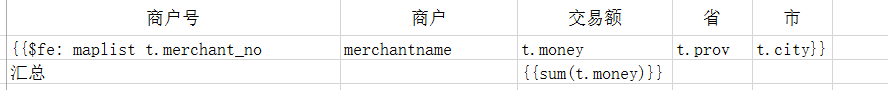
用来解决横向纵向双向遍历统计的问题

#### SUM的设计思路

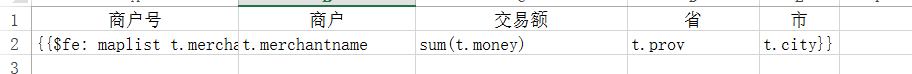
因为模板中的SUM具有位置和属性的双重不确定性,所以针对SUM的这种运算,打算使用记忆的方法来预先记录SUM的位置以及需要SUM的对象,等待遍历处理的时候后置处理了,也防止遍历后,再扫描整个界面来处理问题

SUM的位置可以在FE的上面也可以在FE的下面,也可以直接在FE中来直接SUM比如

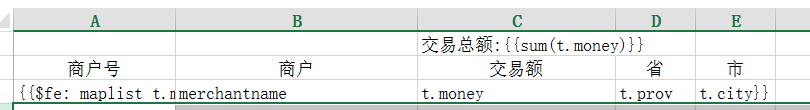
在FE后



在FE中



在FE前



为了防止效率的浪费,回预先记录这个SUM的位置,用了有效处理整个SUM后期处理寻址的问题,在使用中SUM只能SUM list遍历中的数据,不能SUM其他的数据,位置可以自己定义,没有要求

#### 数据写入：

直接在代码中循环为函数写入值

List<Map<String, Object>> mapList= new ArrayList<Map<String, Object>>();

for (int i=0;i<=4;i++) {  
 Map<String, Object> testMap = new HashMap<String, Object>();  
 testMap.put("merchantname", “24325”+i);  
 testMap.put("money", “100”+i);  
 testMap.put("prov", “湖北”);  
 testMap.put("city", “武汉”);  
 mapList.add(testMap);  
}  
map.put("maplist", mapList);

#### 导出：

指定输出路径：modulePath进行导出

TemplateExportParams params = new TemplateExportParams(modulePath, sheetNum);  
Workbook workbook = ExcelExportUtil.*exportExcel*(params, mapList);