Object Excel Mapping 报表映射使用文档

结合<excelDemo>工程，阅读文档。

[Object Excel Mapping 报表映射使用文档 1](#_Toc394064350)

[结合excelDemo工程，阅读文档。 1](#_Toc394064351)

[1. 简单映射 2](#_Toc394064352)

[1.5. 使用@ ColumnCfg标注在属性，get方法，set方法上。 2](#_Toc394064353)

[1.6. 使用OEMContext类 2](#_Toc394064354)

[2. 复杂映射 3](#_Toc394064355)

[2.1. Date类型的复杂映射 3](#_Toc394064356)

[2.2. Integer类型的复杂映射 3](#_Toc394064357)

[2.3. String类型的复杂映射 3](#_Toc394064358)

[2.4. Boolean类型的复杂映射 3](#_Toc394064359)

[2.5. 其他类型标记了map=true 3](#_Toc394064360)

[3. 联合映射的属性 4](#_Toc394064361)

[4. 联合映射 4](#_Toc394064362)

[4.1. 包含关系 4](#_Toc394064363)

[4.1.1. 外层类。 4](#_Toc394064364)

[4.1.2. 内层类。 4](#_Toc394064365)

[4.1.3. 内层类的内层类 4](#_Toc394064366)

[4.1.4. 内层类的外层类 4](#_Toc394064367)

[4.1.5. 被多次包含的类 5](#_Toc394064368)

[4.3. 聚合关系 5](#_Toc394064369)

1. **简单映射**
   1. 使用映射时，对象声明不能使用基础类型(int,short,boolean…)，必须使用Integer,Short,Boolean….包装类对象。
   2. 支持的映射类型Integer，Long，Short，Float，Double，Boolean，Date，String，StringBuffer, Timestamp和含有上述类型的自定义类。
   3. 当从excel中读取纯数字到String中，并且excel单元格类型设置为常规时。n.0会转换成整数n存到String类型中。例如: excel常规为1.1转换为String时 仍是”1.1”，当excel常规1.0转换为string是 ”1”并非是”1.0”
   4. 所有方法必须添加public get属性() public set属性()方法。
   5. 使用@SheetCfg标注在类声明上方，可不填写属性。
      1. 选填属性，title标题默认空，sheetName面板名称默认sheet1，startRowNumber开始行数默认1，sheetId面板标识编号默认-2.
   6. **使用@ ColumnCfg标注在属性，get方法，set方法上。**
      1. 必填属性：Location所在列，0代表第一列。Name列从startRowNumber开开始第一个所填的列名
      2. 标注在属性上代表，即从object生成excel，又从excel读取数据生成object
      3. 标注在get方法上，只从object生成excel
      4. 标注在set方法上，只从excel读取生成object
   7. **使用OEMContext类**
      1. public final static int OFFICE03=3;public final static int OFFICE07=7;两个常亮，用来区分office版本。
      2. public static <E> void generateExecl(List<E> list,OutputStream out,int officeVersion)方法，泛型E做了@SheetCfg注释，传入泛型类的list，和目标OutputStream，直接将list<E>类生成文档到outputstream，officeVersion传入1.7.1常量，使用对应office版本。
      3. public static <E> void generateExecl(List<E> list,String path,String name)，list同上，只是传入本地文件路径path，和文件name（带后缀,根据后缀自动使用office03或07），文档自动生成到该位置。
      4. public static <T> List<T> readExecl(Class<T> clazz,InputStream in,int officeVersion)，Class<T> clazz传入 excel要转化的 类.class。InputStream in为文件读取流，officeVersion传入1.7.1常量，使用对应office版本。
      5. public static <T> List<T> readExecl(Class<T> clazz,String path,String name) Class<T> clazz传入 excel要转化的 类.class。传入本地文件路径path，和文件name（带后缀，根据后缀自动使用office03或07），文档自动生成到该位置。

简单映射的例子见quickstart

1. **复杂映射**
   1. Date类型的复杂映射

@ColumnCfg(location=1,name=”…",map=true,simpleDateFormat="yyyy|MM|dd hh/mm/ss")。让Date类型按simpleDateFormat格式转换成String存入excel并把相同格式的String转换为Date放入该属性

* 1. Integer类型的复杂映射

@ColumnCfg(location=1,name="... ",map=true,mapString="1:正常,2:删除,3:异常,4:其他")。

当map=true时让Integer类型根据mapString转换成对应文字。mapString的格式为 mapString=“数字:文字内容,其他数字:文字内容” 。注意！数字，文字内容不能重复。重复转化回去的时候无法识别value-key。会抛ExcelConstructException。

* 1. String类型的复杂映射

@ColumnCfg(location=2,name="…",map=true,mapString="成功:导出excel表成功,失败:导出excle表失败")

当map=true时将String类型根据mapString转换成对应文字。mapString的格式为 mapString=“文字:文字内容,其他文字:文字内容” 。注意！所有左面文字内容不能重复，所有右面文字不能重复。

* 1. Boolean类型的复杂映射

@ColumnCfg(location=3,name="…",map=true,mapString="#true:导出成功,#false:导出失败")

当map=true时将Boolean类型根据mapString转换成对应文字。mapString的格式为 mapString=“#true:文字内容,#false:文字内容” 。注意！真必须用#true，假必须用#false映射。

* 1. 其他类型标记了map=true

将抛出ExcelConstructException异常

复杂映射的例子见complex

1. **联合映射的属性**
   1. @SheetCfg的属性sheetId用来标识唯一表。如果sheetId重复会错乱。在每一个想生成表的类上标识@SheetCfg(sheetId=”数字”....)数字不能是-1,-2（已被框架使用）。
   2. 注意！同一对象不能在同一sheetId中包含多次。
   3. @ColumnCfg(location=-1,name="")location=-1表示内部包含。只能在属性上标记！在方法上标记运行报错！无其他属性功能。
   4. @ColumnCfgs用于标记多个@ColumnCfg。格式如下

@ColumnCfgs({

@ColumnCfg(location=9,sheetId=1,name="我是内部类的一个属性我在sheetId=1的表单中,location=9"),

@ColumnCfg(location=5,sheetId=2,name="我是同一个内部类的另一个@ColumnCfg，映射在不同的外层类的表单中sheetId=2，位于不同的location=5")

})

1. **联合映射**
   1. **包含关系**
      1. 外层类。
         1. 最外层类需标识@SheetCfg(sheetId=”数字”....)
         2. 在外层类中的自定义对象属性上标识@ColumnCfg(location=-1,name="")
      2. 内层类。
         1. 内层类，及外层所包含的声明的自定义类无需标识@sheetCfg
         2. 在内层类中，使用@ColumnCfgs({@ColumnCfg(location=9,sheetId=1,name="我是内部类的一个属性我在sheetId=1的表单中,location=9")})注释，sheetId标识所属表单的sheetId。Location标识位于该sheetId表单中的列位置。
         3. 约定！所有的内层类的属性、get、set的annotation都使用@ColumnCfgs(@ColumnCfg(sheetId...)...)表示，虽然当只有一个@ColumnCfg时使用@ColumnCfg(sheetId...)仍然工作，但遵守该约定可使代码更明晰。
         4. 内层类与外层类location不能重复。Location表示该列做在位置。
      3. 内层类的内层类
         1. 内层类仍可包含内层类。其所有性质与内层类相同。
         2. 在自定义的属性内部的内部的对象上标记@ColumnCfg(location=-1,name="")
      4. 内层类的外层类
         1. 约定！与5.4.2.3相似。所有的做为外层类使用的属性、get、set的annotation都使用@ColumnCfg(sheetId=...)虽然@ColumnCfgs(@ColumnCfg(sheetId...))时仍然工作，但遵守该约定可使代码更明晰。
         2. 约定！与5.4.2.3.相同。所有的做为内层类使用的属性、get、set的annotation都使用@ColumnCfgs(@ColumnCfg(sheetId...)...)表示
      5. 被多次包含的类
         1. 在内层类标记@ColumnCfgs包含多个@ColumnCfg分配不同的sheetId

例如@ColumnCfgs({

@ColumnCfg(location=9,sheetId=1,name="… "),

@ColumnCfg(location=5,sheetId=2,name="xxx"),

….

})

* 1. 例子见Include
  2. **聚合关系**
     1. 在同一个外层类中包含多个内层类。分别在属性对象上标记@ColumnCfg(location=-1,name="")location=-1
     2. Location不能重复
     3. 内层类仍可包含内层类。
  3. 例子见union