**狮桥研发技术文档**

**2016年3月(编写日期)**

**V1.0(版本号)**

关于本文档

对于本文档的版权与保密信息的描述。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 主 题 | | 项目结构及技术使用规范 | | | | | | |
| 说　　明 | |  | | | | | | |
| 适用对象 | | 系统研发人员 | | | | | | |
| 修订历史 | | | | | | | | |
| 版本 | 章节 | | 类型 | 修订日期 | 修订者 | 复核日期 | 复核人 | 修订说明 |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |

**说明：类型－创建（C）、修改（U）、删除（D）、增加（A）；**

目录

[关于本文档 2](#_Toc460333349)

[1. 前言 4](#_Toc460333350)

[1.1 编写目的 4](#_Toc460333351)

[1.2 适用范围 5](#_Toc460333352)

[1.3 术语定义 5](#_Toc460333353)

[1.4 参考文档 5](#_Toc460333354)

[2. 项目结构 5](#_Toc460333355)

[2.1 LB\_BASE 5](#_Toc460333356)

[2.2 FL\_INTERFACE 6](#_Toc460333357)

[2.3 FL\_SERVICE 7](#_Toc460333358)

[2.4 FL\_WEB 10](#_Toc460333359)

[2.5 结构图 10](#_Toc460333360)

[3. 项目配置 11](#_Toc460333361)

[3.1 MAVEN 11](#_Toc460333362)

[3.2 DUBBO 12](#_Toc460333363)

[3.2.1 说明 12](#_Toc460333364)

[3.3 项目依赖 12](#_Toc460333365)

[4. 定时器使用 13](#_Toc460333366)

[4.1 使用说明 13](#_Toc460333367)

[4.2 配置步骤 13](#_Toc460333368)

[5. 命名规范 13](#_Toc460333369)

[5.1 类方法命名规范说明 13](#_Toc460333370)

[5.2 配置文件命名规范说明 14](#_Toc460333371)

[5.2.1 DUBBO 14](#_Toc460333372)

[5.2.2 MABATIS 14](#_Toc460333373)

[5.2.3 PROPERTIES 14](#_Toc460333374)

[5.2.4 QUARTZ 14](#_Toc460333375)

[5.2.5 XML 14](#_Toc460333376)

[5.2.6 FILE 15](#_Toc460333377)

[5.3 包命名规范说明 15](#_Toc460333378)

[5.4 JSP命名规范 15](#_Toc460333379)

[5.5 其它规范 15](#_Toc460333380)

[5.5.1 暴露DUBBO API接口 15](#_Toc460333381)

[5.5.2 内部接口实现类 16](#_Toc460333382)

[5.5.3 接口和方法注释 16](#_Toc460333383)

[5.5.4 实体类注释 17](#_Toc460333384)

[6. 常用工具 17](#_Toc460333385)

[6.1 使用说明 17](#_Toc460333386)

[6.1.1 CollectionUtils 17](#_Toc460333387)

[6.1.2 DateUtils 17](#_Toc460333388)

[6.1.3 DecimalUtils 17](#_Toc460333389)

[6.1.4 FileUtils 17](#_Toc460333390)

[6.1.5 HttpClientUtils 17](#_Toc460333391)

[6.1.6 PropertiesUtils 18](#_Toc460333392)

[6.1.7 ReflectHelper 18](#_Toc460333393)

[6.1.8 StringUtils 18](#_Toc460333394)

[6.1.9 SysLogUtils 18](#_Toc460333395)

[7. 研发代码步骤(以Demo为例子) 19](#_Toc460333396)

[8. 代码生成工具 19](#_Toc460333397)

[8.1 使用说明 19](#_Toc460333398)

[8.2 使用注意事项 20](#_Toc460333399)

# 前言

## 编写目的

有效管理项目结构及技术规范,防止研发过程中造成结构混乱，技术混乱现象以及影响项目稳定性、扩展性等。

由以下项目组成：

1. LB\_BASE为公共类及公共工具的定义。
2. FL\_INTERFACE为公共暴露接口及常量类的定义。
3. FL\_SERVICE 为服务支撑项目，其中定义持久层相关文件及对外接口实现等。
4. FL\_WEB 为客户端项目，其中定义数据转换实现和所需文件。

## 适用范围

本文档适用读者包括以下人员：

* 项目PMO
* 系统研发组成员

## 术语定义

PO : 持久层对象

DTO: 服务端与客户端传递数据的对象

VO: 客户端展示数据使用对象

DAO: 持久层

SERVICE: 服务层 也是服务提供者

CONTROLLER: REST控制层

BASE: 基类

TRANSFORM: 数据转换类

## 参考文档

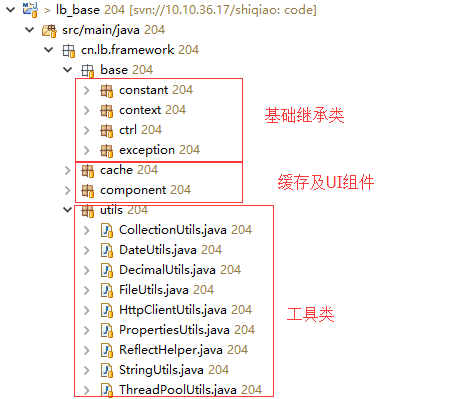
无.

# 项目结构

## LB\_BASE

LB\_BASE项目中包括公共使用的全局常量类、控制层基类、异常类、工具类、缓存(EhCache、Redis), 所有研发人员要熟悉其中各类在什么时候引用, 研发人员不能随意改动LB\_BASE项目中的任何代码,如果有公共需要请提出相应的需求。

结构如下：



注：缓存配置文件在fl\_service、fl\_web项目中均有配置,在使用过程中需要定义这两个项目中的文件。

## FL\_INTERFACE

FL\_INTERFACE 项目是客户端和服务端所拥有共同类及接口定义。

由各个业务模块的常量类、DTO类、查询对象、暴露接口、分页类,如果需要建立暴露接口、DTO、常量类 ,按照demo模块自定义即可。

1、自定义类名称规范注意事项：

暴露服务接口：I + 模块名称 + ApiService 例：IDemoApiService

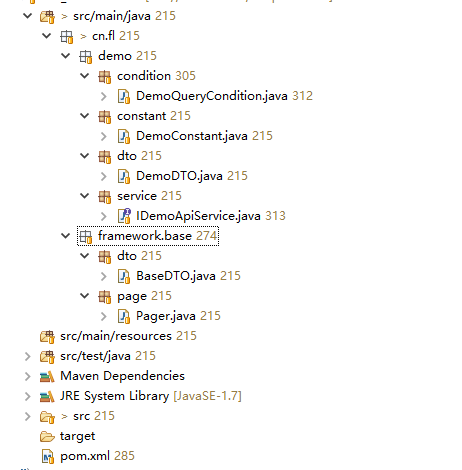
DTO : 实体类名称 + DTO 继承 BaseDTO 例：DemoDTO

常量类： 模块名称+Constant 或 实体名称+Constant 例：DemoConstant

查询对象: 模块名称 + QueryCondition 例：DemoQueryCondition 或者

模块名称 + QC 例：DemoQC

结构如下：



## FL\_SERVICE

FL\_SERVICE项目为客户端提供业务支撑(同时暴露Dubbo接口)。

其中由 DAO 、 PO 、 SERVICE、TRANSFORM 、FRAMEWORK(研发勿动)定义以及配置文件 DUBBO、MYBATIS、PROPERTIES、QUARZ、SPRING、XML 组成，研发人员可按照demo模块进行配置开发 。

1、自定义类名称规范注意事项：

DAO: 实体类名称 + Mapper 继承IBaseDao 例：DemoMapper

PO: 实体类名称 例：Demo

SERVICE: I + 实体类名称 + Service 例：IDemoService

SERVICE.IMPL：实体类名称 + ServiceImpl 例：DemoServiceImpl

暴露接口实现：模块名称 + ApiServiceImpl 例：DemoApiServiceImpl

TRANSFORM: 实体类名称 + Transform 例：DemoTransform

参照Demo模块类继承模式。

2、自定义配置文件名称规范注意事项(名称小写)：

DUBBO: spring-dubbo-provider- + 实体类名称(多名词要用中划线分割) + .xml 例：spring-dubbo-provider-demo.xml

MYBATIS: 实体类名称 + -result-mapper.xml

例：demo--result-mapper.xml

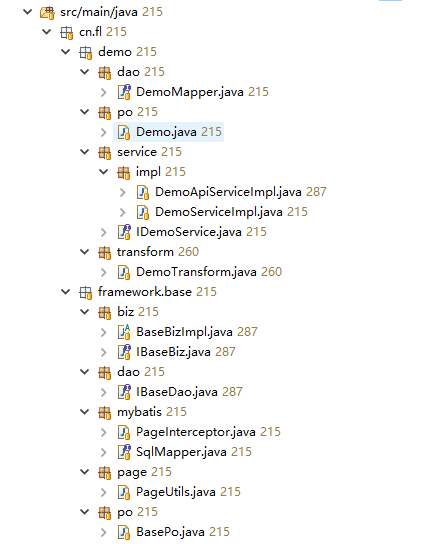
实体类名称 + -mapper.xml

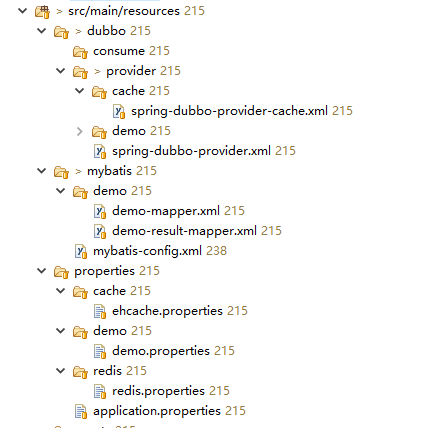
例：demo-mapper.xml

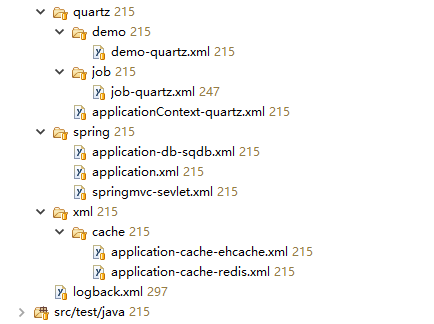
PROPERTIES：模块名 + .properties 例：demo.properties

QUARZ：模块名 + -quartz.xml 例：demo-quartz.xml

结构如下：







## FL\_WEB

FL\_WEB为客户端(DUBBO消费方) , 需要请求FL\_SERVICE服务端(DUBBO 服务提供方)请求业务数据。

其中由CONTROLLER 、SERVICE、TRANSFORM、VO、FRAMEWORK及配置文件组成。研发人员可按照demo模块进行开发 。

1、自定义类名称规范注意事项：

CONTROLLER：实体类名称 + Controller 或 模块名 + Controller

例：DemoController

SERVICE: I + 实体类名称 + Service 例： IDemoService

SERVICE.IMPL: 实体类名称 + ServiceImpl 例：DemoServiceImpl

TRANSFER: 实体类名称 + VOTransform 例：DemoVOTransform

VO: 实体类名称 + VO 例：DemoVO

参照Demo模块类继承模式。

2、自定义配置文件名称规范注意事项(名称小写)：

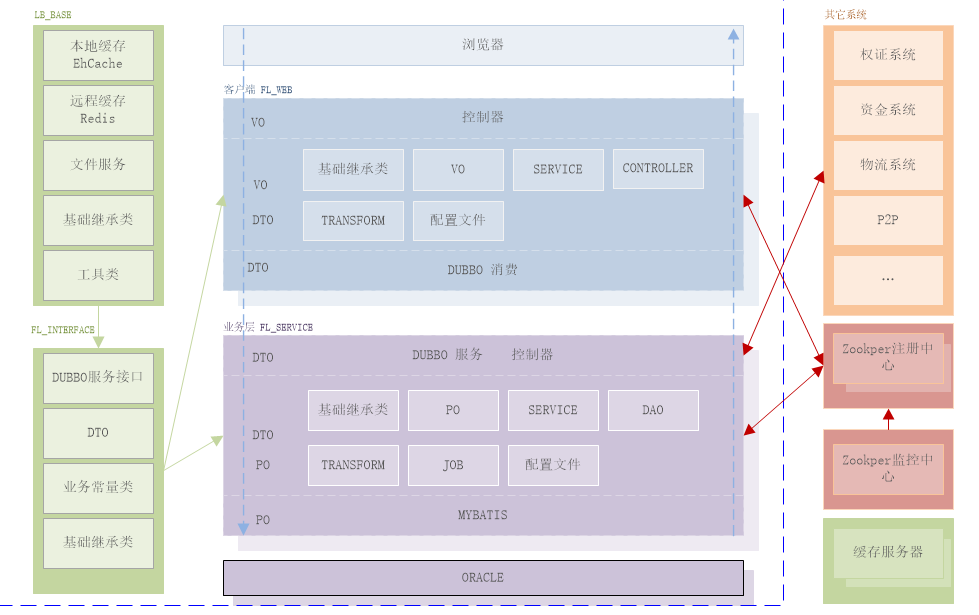
DUBBO: spring-dubbo-consume-实体类名称.xml

例：spring-dubbo-consume-demo.xml

PROPERTIES：模块名 + .properties 例：demo.properties

QUARZ：模块名 + -quartz.xml 例：demo-quartz.xml

## 结构图





# 项目配置

## MAVEN

项目统一由Maven管理, 结构为：

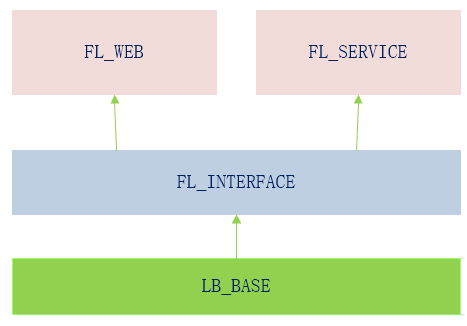
fl-root  
    　　 |----pom.xml  
    　　 |----fl-web  
       |----pom.xml  
    　　 |----fl-service  
        　　 |----pom.xml  
    　 |----fl-interface  
       |----pom.xml  
    　　 |----lb-base  
         |----pom.xml

## DUBBO

### 说明

1. DUBBO 配置分为服务提供方(FL\_SERVICE)和消费方(FL\_WEB)
2. 服务方主配置在resources/dubbo/provider/spring-dubbo-provider.xml中。
3. 消费方主配置在resources/dubbo/consume/spring-dubbo-consume.xml中。
4. 各业务模块需在同级目录下建立模块文件夹后建立属于自己的服务xml,参照demo模块.

## 项目依赖



# 定时器使用

## 使用说明

如果业务需要定时器做某些操作时可以考虑使用quartz定时器.

Quartz采用无数据库模式配置。

## 配置步骤

1. 定义好需要执行的类及方法,可以参考系统级定时器配置/fl\_service/src/main/java/cn/fl/system/job/IJobService.java
2. 在/fl\_service/src/main/resources/quartz 目录下建立属于自己的模块,参照job-quartz.xml 配置xml文件，命名规则必须是 \*-quzrtz.xml , 内容请参考job-quartz.xml内容,什么时候执行根据业务需要自行配置。
3. 把\*-quartz.xml中\*TaskTime 名字放到applicationContext-quartz.xml中startQuertz标签list下面即可。

# 命名规范

## 类方法命名规范说明

1、所有方法名称均采用驼峰命名法。

2、Mapper 方法命名：新增(insert\*) 、删除(delete\*)、修改(update\*)、查询(select\* find\* get\*)、分页查询(select\*Page)。

3、Service 方法命名：新增(add\*) 、删除(remove\*)、修改(modify\*)、查询(select\* get\*)、分页查询(search\*Page)。

4、类名以驼峰命名法且首字母必须大写。

5、所有接口方法要抛出BaseException运行时异常, 根据业务处理逻辑对错误使用异常模式抛出,比如参数为空, 则 throw new BaseException(BaseExceptionConstant. BASE\_EXCEPTION\_CODE\_600,

具体描述信息);

## 配置文件命名规范说明

### DUBBO

1、消费方配置文件都配置在resources/dubbo/consume下以模块名称(小写)区分, 自定义文件(均小写)格式：spring-dubbo-consume-模块名称.xml。

2、服务方配置文件都配置在resources/dubbo/provider下以模块名称(小写)区分,自定义文件(均小写)格式: spring-dubbo-provider-模块名称.xml

### MABATIS

1、所有SQL配置文件都在resources/mybatis目录下定义自己的模块(小写), 每个实体对应两个文件 \*-mapper 与 \*-result-mapper 其中\*为实体名称以名词分割后用中划线连接的字符串(小写),如：demo-demo-mapper.xml demo-demo-result-mapper

### PROPERTIES

1、所有属性配置文件都在resources/properties目录下定义自己的模块(小写), 属性文件名称定义为小写, 其中内部命名要严格按照：模块名.文件名.properties.\* 格式, 如果命名不规范就会在全局缓存中不存在数据的时候直接查询文件失败。如：cache.ehcache.properties.local.is.open

### QUARTZ

1、所有属性配置文件都在resources/quartz目录下定义自己的模块(小写), 文件名称定义为小写, 格式：\*-quartz.xml。具体内容参考job-quartz.xml.

### XML

1、所有属性配置文件都在resources/xml目录下定义自己的模块(小写), 文件名称定义为小写, 格式：application-模块-自定义.xml。

### FILE

1、所有文件存放位置都在resources/file目录下定义自己的模块(小写), 将文件放入即可。

## 包命名规范说明

1. 所有包在cn.fl目录下,如果各业务需要分别建立属于自己的模块包,包名必须全部为小写, 模块包下建立po 、vo、dto等包即可。

## JSP命名规范

**大模块名+小模块名+关键词(如下) ，命名均已驼峰命名法, 如果名称过长可以对单独词语进行缩写(如system 可以缩写为sys, customer可以缩写为cust ),但是名称一定要有明确的含义。**

**1、增加 add**

**2、删除 remove delete del**

**3、修改 update modify**

**4、查询列表 list**

**5、管理 manage**

**6、设置 setting**

**如客户:**

**新增： custCstBscInfoAdd**

**列表页：custCstBscInfoList**

**修改：custCstBacInfoModify**

**管理页面：custCstBacInfoManage**

## 其它规范

### 暴露DUBBO API接口

1、所有接口必须抛出throws BaseException 运行时异常。

2、实现类中只能调用其它服务层接口获取数据。

3、类中不允许出现非DTO对象进行数据传输, 如 PO VO。

4、类中不允许出现复杂业务逻辑,只能做基本的验证操作、异常处理、操作日志。所有的异常均已throw new BaseException(BaseExceptionConstant. BASE\_EXCEPTION\_CODE\_600,

具体错误信息描述)模式向上抛出。

### 内部接口实现类

1、所有接口必须抛出throws BaseException 运行时异常。

2、BaseBizImpl 中已经实现了基本方法如下：

public Long add(DTO entity) throws BaseException;

public boolean modify(DTO entity) throws BaseException;

public DTO selectById(Long id) throws BaseException;

public Pager searchListPage(Pager page) throws BaseException;

public Pager searchListPage(Pager page) throws BaseException;

如果有需要可以重写本方法,或者自定方法即可,

1. 内部接口绝对不可以直接调用其它服务层的Mappr服务,只能调用Service接口获取数据, 返回的对象不能包含PO对象。PO对象只能在自己的服务中使用。

### 接口和方法注释

1、接口类要明确写上是什么接口及用途说明。

2、方法体内要处理什么业务要在接口中写明。

3、方法体中复杂逻辑一定要标明处理思路及业务逻辑说明。

4、修改接口和方法时要同时更新注释。

### 实体类注释

1、注明实体类所属对象描述。

2、所有属性和get set方法必须写明字段说明,这样有助于调用者在调用的时候明确看到属性是干什么的, 也同时避免在很多字段的情况下自己还得去和模型或者数据库对比含义。

7、查询对象命名为：模块名 + QueryCondition 或模块名 + QC

# 常用工具

## JAVA使用说明

项目中如果用到与工具类相关的验证等务必使用工具类.

### CollectionUtils

对集合空验证和非空的验证。

### DateUtils

对时间的转换以及时间的获取等。

### DecimalUtils

对计算的加减乘除的处理，程序中所有Double类型的加减乘除运算必须要使用本工具处理。

### FileUtils

对文件的简单处理。

### HttpClientUtils

对发送POST请求工具类,根据参数配置好参数后使用。

### PropertiesUtils

项目中所有属性文件获取均通过本类获取。

属性获取方式是先到缓存中获取,如果缓存中不存在,则按照key名称格式拆分后组合文件路径及文件名称获取属性文件,然后获取到key之后把属性文件加载到缓存中,如果key的名称不按照命名规范严格编写就可能造成获取不到数据的错误。

### ReflectHelper

反射类。

### StringUtils

字符串工具类，对字符串的判空、 邮箱格式匹配、是否是整数、是否是手机号。

### SysLogUtils

系统日志工具类, FL\_WEB、FL\_SERVICE 根据业务逻辑要加入操作日志,使用方法如下：

SysLogUtils.*info*(***logger***, "toListPage", "说明", CurrentThreadContext.*getCurrentUserName*());

SysLogUtils.*error*(***logger***, "toListPage", "说明", CurrentThreadContext.*getCurrentUserName*());

参数1：当前类log定义

参数2：方法名称

参数3：操作说明

参数4：操作人(如果是FL\_WEB一定要加,如果在FL\_SERVICE中拿不到操作人则使用没有操作人的重载方法)

## JS使用说明

### CollectionUtils

### DateUtils

### LogUtils

### NumberUtils

### StringUtils

### UIUtils

定义统一公共使用的JS方法,如：

成功提示

错误提示

警告提示

# 研发代码步骤(以Demo为例子)

参考FL\_SERVICE DEMO模块

FL\_WEB DEMO 模块

# 代码生成工具

## 使用说明

1、FL\_GEN\_CODE 项目会为各位研发节省很多配置时间.

2、工具会生成文件目录如下：

constant 、controller、dao 、dto、mybatis、po、service、transform、vo、web

3、在/fl\_gen\_code/config/config.properties 中配置数据库连接及用户名密码

4、在CodeGen中main方法中配置数据库命名空间名称(大写)和所研发模块(小写), 如命名空间名称为ORCL , 所研发负责模块为user , 则配置为：

initParameter("ORCL","user"); ORCL所在的数据库中只能存在属于user模块要生成的表, 这样可以清晰的查看及配置相关文件。

5、Run As 执行方法后会在项目根目录中生成gen目录存放文件。

6、各个文件存放位置：

constant 、dto 、service暴露服务接口(自行定义)、 存在放FL\_INTERFACE项目中。

po、dao、mybatis(不包括\* -mybatis-config.xml)、transform(不包括VO传输对象) 、service存放FL\_SERVICE项目中，其中mybatis别名需要到resources/mybatis/mybatis-config.xml别名区域中参考demo配置。

vo、 transform(只包含VO)、web下service、controller存在放FL\_WEB项目中

7、配置暴露接口文件,一般情况下一个模块下有一个暴露接口即可,到FL\_INTERFACE 配置接口,如user模块配置接口为 IUserApiService(无需继承), 在里面定义暴露的方法即可,如果模块内容叫多则要分多个API接口, 每个API接口中的方法建议不要超过30个,接口定义要尽量通用。

8、 实现暴露接口, 到FL\_SERVICE项目中如user模块下service.impl中实现暴露接口 UserApiServiceImpl , 实现IUserApiService的方法获取数据都要依赖于模块下未暴露的接口。类为Spring bean 加上注解 @Service

9、配置DUBBO 暴露接口服务文件, 到 FL\_SERVICE项目中resources/dubbo/provider目录下建立模块文件夹如 user ,创建spring-dubbo-provider-user.xml文件，内容参考demo目录。

10、配置DUBBO 接口消费方文件, 到FL\_WEB项目中resources/dubbo/consume 目录下建立模块文件夹如 user , 创建spring-dubbo-consume-user.xml , 内容参考demo目录。

11、一定参照5.2 使用注意事项检查PO实体及转换类DTO VO 字段类型是否是数据库原始意义字段。

12、最后编写FL\_WEB、 FL\_SERVICE中的方法和处理业务逻辑就可以飞起来了。

## 使用注意事项

1. 工具中对NUMBER数据库字段类型控制分为以下几种：

长度为1 且小数位为0的定义 java类中定义为Integer类型。

长度为4 且小数位为0的定义 java类中定义为Integer类型。

长度为未设置且小数位未设置 java 类型定义为Long类型。

长度设置不限且小数位数大于0 java 类型定义为Double类型。

字段完全匹配”ID” 的为Long类型

字段后缀完全匹配”\_ID”的为Long类型

字段后缀完全匹配”\_AMT” 或者 “\_PCT” 的为Double类型

字段后最完全匹配”\_QTY” 的为Integer类型

在java生成PO、 VO、 DTO文件后要严格检查匹配字段类型,如果有问题自行修改。

2、PO 、VO 、DTO中的主键、创建人、创建时间、修改人、修改时间都是内置在对象中的,不要重复定义这些字段。