万信金融P2P项目编码规范

一. 命名规约

1. **【强制】禁止拼音缩写，避免阅读者费劲猜测；尽量不用拼音，除非中国式业务词汇没有通用易懂的英文对应**

```
禁止： DZ[打折] / getPFByName()[评分]
尽量避免：Dazhe / DaZhePrice
```

2. **【推荐】禁止使用非标准的英文缩写**

```
反例： AbstractClass 缩写成 AbsClass；condition 缩写成 condi
```

3. **【强制】禁用其他编程语言风格的前缀和后缀**

在其它编程语言中使用的特殊前缀或后缀，如`_name`，`name_`，`mName`，`i_name`，在Java中都不建议使用

4. **【推荐】包名全部小写点分隔符之间尽量只有一个英语单词，即使有多个单词也不使用下划线或大小写分隔**

```
正例： cn.itcast.javatool
反例： cn.itcast.java_tool, cn.itcast.javaTool
```

5. **【强制】类名与接口名使用UpperCamelCase风格，遵从驼峰形式**

Tcp，Xml等缩写也遵循驼峰形式，可约定例外如：DTO/ VO等

```
正例：UserId / XmlService / TcpUdpDeal / UserVO
反例：UserID / XMLService / TCPUDPDeal / UserVo
```

6. **【强制】方法名、参数名、成员变量、局部变量使用lowerCamelCase风格，遵从驼峰形式**

```
正例： localValue / getHttpMessage();
```

7. **【强制】常量命名全大写，单词间用下划线隔开力求语义表达完整清楚，不要嫌名字长**

```
正例： MAX_STOCK_COUNT
反例： MAX_COUNT
```

例外：当一个static　final字段不是一个真正常量，比如不是基本类型时，不需要使用大写命名

```
private static final Logger logger = Logger.getLogger(MyClass.class);
```

8. **【推荐】枚举类名以Enum结尾；抽象类使用Abstract或Base开头；异常类使用Exception结尾；测试类以它要测试的类名开始，以Test结尾**

> 正例：DealStatusEnum， AbstractView，BaseView， TimeoutException，UserServiceTest

9. **【推荐】实现类尽量用Impl的后缀与接口关联，除了形容能力的接口**

> 正例：CacheServiceImpl 实现 CacheService接
> 口正例：Foo 实现 Translatable接口

10. **【强制】POJO类中布尔类型的变量名，不要加is前缀，否则部分框架解析会引起序列化错误**

> 反例：Boolean isSuccess的成员变量，它的GET方法也是isSuccess()，部分框架在反射解析的时候，“以为”对应的成员变量名称是success，导致出错

11. **【推荐】各层命名规约**

DAO 层方法命名规约

1) 获取单个对象的方法用 select 做前缀

2) 插入的方法用insert做前缀

3) 删除的方法用delete做前缀

4) 修改的方法用update做前缀

Service 层方法命名规约

1) 获取单个对象的方法用get做前缀

2) 获取多个对象的方法用 query 做前缀

3) 插入的方法用create做前缀

4) 删除的方法用remove做前缀

5) 修改的方法用modify做前缀

12. **【强制】领域分层规约**

分层领域模型规约：

DO（ Data Object ）：与数据库表结构一一对应，通过DAO层向上传输数据源对象。
DTO（ Data Transfer Object ）：数据传输对象，Service或Manager向外传输的对象。
BO（ Business Object ）：业务对象。 由Service层输出的封装业务逻辑的对象。
VO（ View Object ）：显示层对象，通常是Web向模板渲染引擎层传输的对象。
POJO（ Plain Ordinary Java Object ）：在本手册中， POJO专指只有setter/getter/toString的简单类，包括DO/DTO/BO/VO等。
Query：数据查询对象，各层接收上层的查询请求。

二. 常量定义

1. **【推荐】不要使用一个常量类维护所有常量，要按常量功能进行归类，分开维护 说明:大而全的常量类，杂乱无章，使用查找功能才能定位到修改的常量，不利于理解和维护**

   正例:缓存相关常量放在类 CacheConsts 下;系统配置相关常量放在类 ConfigConsts 下

三. 注释规约

1. **【推荐】基本的注释要求**

   完全没有注释的大段代码对于阅读者形同天书，注释是给自己看的，即使隔很长时间，也能清晰理解当时的思路；注释也是给继任者看的，使其能够快速接替自己的工作

   代码将被大量后续维护，注释如果对阅读者有帮助，不要吝啬在注释上花费的时间(但也综合参见规则2，3)

   第一、能够准确反应设计思想和代码逻辑；第二、能够描述业务含义，使别的程序员能够迅速了解到代码背后的信息

   除了特别清晰的类，都尽量编写类级别注释，说明类的目的和使用方法

   除了特别清晰的方法，对外提供的公有方法，抽象类的方法，同样尽量清晰的描述：期待的输入，对应的输出，错误的处理和返回码，以及可能抛出的异常

2. **【推荐】通过更清晰的代码来避免注释**

   在编写注释前，考虑是否可以通过更好的命名，更清晰的代码结构，更好的函数和变量的抽取，让代码不言自明，此时不需要额外的注释

3. **【强制】代码修改的同时，注释也要进行相应的修改尤其是参数、返回值、异常、核心逻辑等的修改**

4. **【推荐】注释不要为了英文而英文**

   如果没有国际化要求，中文能表达得更清晰时还是用中文

5. **【推荐】TODO标记，清晰说明代办事项和处理人**

6. **【推荐】合理处理注释掉的代码**

   如果后续会恢复此段代码，在目标代码上方用 `///` 说明注释动机，而不是简单的注释掉代码

四. 方法设计

1. **【推荐】方法的长度度量**

   方法尽量不要超过100行

2. **【推荐】尽量减少重复的代码，抽取方法**

   超过5行以上重复的代码，都可以考虑抽取公用的方法

3. **【推荐】方法参数最好不超过3个，最多不超过7个**

   1）如果多个参数同属于一个对象，直接传递对象

   例外: 你不希望依赖整个对象，传播了类之间的依赖性

   2）将多个参数合并为一个新创建的逻辑对象

   例外: 多个参数之间毫无逻辑关联

   3）将函数拆分成多个函数，让每个函数所需的参数减少

4. **【推荐】不使用不稳定方法，如com.sun.*包下的类，底层类库中internal包下的类** `com.sun.*` ，`sun.*` 包

下的类，或者底层类库中名称为internal的包下的类，都是不对外暴露的，可随时被改变的不稳定类

五. URL规约

采用Restful风格的URL

1. 常用的HTTP动词有下面五个（括号里是对应的SQL命令）

```
GET（SELECT）：从服务器取出资源（一项或多项）
POST（CREATE）：在服务器新建一个资源
PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）
PATCH（UPDATE）：在服务器更新资源（客户端提供改变的属性）
DELETE（DELETE）：从服务器删除资源
```

2. 尽量避免在URL中使用动词

正例：
GET /zoos：列出所有动物园
POST /zoos：新建一个动物园
GET /zoos/ID：获取某个指定动物园的信息
PUT /zoos/ID：更新某个指定动物园的信息（提供该动物园的全部信息）
PATCH /zoos/ID：更新某个指定动物园的信息（提供该动物园的部分信息）
DELETE /zoos/ID：删除某个动物园
GET /zoos/ID/animals：列出某个指定动物园的所有动物
DELETE /zoos/ID/animals/ID：删除某个指定动物园的指定动物

六. MQ规约

包括RocketMQ，RabbitMQ等消息队列产品

1. **全部使用大写，单词之间以'_'分割**
2. **生产组(Producer Group)以'PID'为前缀**
3. **消费组(Consumer Group)以'CID'为前缀**
4. **主题(Topic)以'TP'为前缀**

七. 其他规约

1. **【参考】尽量不要让魔法值（即未经定义的数字或字符串常量）直接出现在代码中**

```
//WRONG
String key = "Id#taobao_"+tradeId;
cache.put(key, value);
```

例外：-1,0,1,2,3 不认为是魔法数

2. **【推荐】变量声明尽量靠近使用的分支**

八. 万信金融项目规约

1. 所有的Entity和DTO都应使用lombok框架的@Data注解，不再自行编写getter/setter方法

```
@Data
@TableName("consumer")
public class Consumer implements Serializable

    { private static final long serialVersionUID =

    1L;

    /**
     * 主键
     */
    @TableId(value = "ID", type = IdType.AUTO)
    private Long id;

    /**
     * 用户名
```

```
    private String username;
    ...
}
```

2. 各服务controller的接口都应定义在wanxinp2p-api项目下cn.itcast.wanxinp2p.api包中

```
package  cn.itcast.wanxinp2p.api.consumer;

public interface ConsumerApi {
  ...
}
```

3. 所有DTO都应定义在wanxinp2p-api项目下cn.itcast.wanxinp2p.api.[serviceName].model中

```
package  cn.itcast.wanxinp2p.api.consumer.model;

@Data
@ApiModel(value = "ConsumerDTO", description = "平台c端用户信息")
public class ConsumerDTO {
    ...
}
```

4. controller中所有方法返回值都应是**RestResponse**(cn.itcast.wanxinp2p.common.domain)

```
    @ApiModelProperty("响应错误编码,0为正常")
    private int code;

    @ApiModelProperty("响应错误信息")
    private String msg;

    @ApiModelProperty("响应内容")
    private T result;
```

其中T为泛型，记录了返回的具体业务对象数据

5. controller中所有方法返回的具体业务对象数据必须是DTO，不能直接用Entity

```
    /**
     * 获取借款人用户信息
     * @return
     */
    RestResponse<BorrowerDTO> getBorrower(Long id);

    /**
     * 根据用户名获取用户信息
     * @param username
     * @return
     */
    RestResponse<ConsumerDTO> getConsumerByUsername(String username);
```

```
    /**
     * 获取当前登录用户
     * @return
     */
    RestResponse<ConsumerDTO>  getMyConsumer();
```

6. service中所有的分页查询返回类型应是**PageVO**(cn.itcast.wanxinp2p.common.domain)

```
    /**
     * 检索用户列表
     * @param consumerQueryDTO 检索参数
     * @param pageNo
     * @param pageSize
     * @param sortB
     * @param order
     * @return
     */
    PageVO<ConsumerDTO> queryConsumers(ConsumerQueryDTO consumerQueryDTO, Integer pageNo,
                                       Integer pageSize, String sortBy, String order);
```

7. URI

(1)受保护c端用户接口
Url格式：/服务名称/my/资源名称/*
方法名格式：[动作]My[资源名称]
访问方式：header中需要携带C端用户认证所获取的Access Token才可访问

(2)受保护的b端用户接口
Url格式：/服务名称/m/资源名称/*
访问方式：header中需要携带B端用户认证所获取的Access Token才可访问

(3)公开资源
Url格式：/服务名称/资源名称/*
方法名格式：[动作][资源名称]
访问方式：无限制

(4)受限资源
Url格式：/服务名称/l/资源名称/*
访问方式：仅供微服务之间调用