## 代码中的命名必须使用统一英文直译，需见名知意。 正例：UserController 反例：yonghuController/aController等

## 代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。 说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式 也要避免采用。 正例：tofobao/ tfb/esgcc等国际通用的名称，可视同英文。 反例：dianfeituoshou / getPingfenByName() [评分] / int 某变量 = 3

## 类名使用 UpperCamelCase 风格，即每个单词首字母大写，后面字母小写.正例：MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion 反例：macroPolo / UserDo / XMLService / TCPUDPDeal / TAPromotion

## 方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从 驼峰形式,即整体首字母小写，每个单词首字母大写。 正例： localValue / getHttpMessage() / inputUserId

## 常量命名全部大写，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：CONTROLLER\_CODE\_ERROR

## 所有POJO类必须进行序列化，通过实现Serializable接口，同时通讯和web相关类必须序列化。

## 包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用 单数形式，但是类名如果有复数含义，类名可以使用复数形式。 正例：应用工具类包名为 com.tofobao.controller、类名为 DateUtils（此规则参考 spring 的框架结构）

## 杜绝完全不规范的缩写，避免望文不知义，任何自定义编程元素在命名时，使用尽量完整的单词 组合来表达其意。 正例：List<User> userList反例：“缩写”命名成 List<User> uList等，此类随 意缩写严重降低了代码的可阅读性。

## 如果模块、接口、类、方法使用了设计模式，在命名时需体现出具体模式。 说明：将设计模式体现在名字中，有利于阅读者快速理解架构设计理念。 正例：public class OrderFactory; public class LoginProxy; public class ResourceObserver;

## 有默认实现，那么这个 default 方法，是对所有实现类都有价值的默 认实现。

## 接口和实现类的命名规则：对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部 的实现类用 Impl 的后缀与接口区别。 正例：CacheServiceImpl 实现 CacheService 接口。

## 各层命名规约： A) Service/DAO 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 list 做前缀，复数形式结尾如：listObjects。 3） 获取统计值的方法用 count 做前缀。 4） 插入的方法用 save/insert 做前缀。 5） 删除的方法用 remove/delete 做前缀。 6） 修改的方法用 update 做前缀。

## 大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行；如果 是非空代码块则： 1） 左大括号前不换行。 2） 左大括号后换行。 3） 右大括号前换行。 4） 右大括号后还有 else 等代码则不换行；表示终止的右大括号后必须换行。

## 左小括号和字符之间不出现空格；同样，右小括号和字符之间也不出现空格；而左大 括号前需要空格。详见第 5 条下方正例提示。 反例：if (空格 a == b 空格)

## if/for/while/switch/do 等保留字与括号之间都必须加空格。

## 任何二目、三目运算符的左右两边都需要加一个空格。 说明：运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号等。

## 所有的覆写方法，必须加@Override 注解。 说明：getObject()与 get0bject()的问题。一个是字母的 O，一个是数字的 0，加@Override 可以准确判断是否覆盖成功。另外，如果在抽象类中对方法签名进行修改，其实现类会马上编 译报错。

## 所有的相同类型的包装类对象之间值的比较，全部使用 equals 方法比较。 说明：对于 Integer var = ? 在-128 至 127 范围内的赋值，Integer 对象是在 IntegerCache.cache 产生，会复用已有对象，这个区间内的 Integer 值可以直接使用==进行 判断，但是这个区间之外的所有数据，都会在堆上产生，并不会复用已有对象，这是一个大坑， 推荐使用 equals 方法进行判断。

## 类内方法定义的顺序依次是：公有方法或保护方法 > 私有方法 > getter/setter 方法。 说明：公有方法是类的调用者和维护者最关心的方法，首屏展示最好；保护方法虽然只是子类 关心，也可能是“模板设计模式”下的核心方法；而私有方法外部一般不需要特别关心，是一个 黑盒实现；因为承载的信息价值较低，所有 Service 和 DAO 的 getter/setter 方法放在类体 最后。

## 对于注释的要求：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。完全没有注释的大段代码对于阅读者形同

## 类、类属性、类方法的注释必须使用 Javadoc 规范，使用/\*\*内容\*/格式，不得使用 // xxx 方式。 说明：在 IDE 编辑窗口中，Javadoc 方式会提示相关注释，生成 Javadoc 可以正确输出相应注 释；在 IDE 中，工程调用方法时，不进入方法即可悬浮提示方法、参数、返回值的意义，提高 阅读效率。

## 所有的抽象方法（包括接口中的方法）必须要用 Javadoc 注释、除了返回值、参数、 异常说明外，还必须指出该方法做什么事情，实现什么功能。 说明：对子类的实现要求，或者调用注意事项，请一并说明。

## 所有的类都必须添加创建者和创建日期。

## 方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释 使用/\* \*/注释，注意与代码对齐。

## 所有的枚举类型字段必须要有注释，说明每个数据项的用途。

## 与其“半吊子”英文来注释，不如用中文注释把问题说清楚。专有名词与关键字保持 英文原文即可。 反例：“TCP 连接超时”解释成“传输控制协议连接超时”，理解反而费脑筋。

## 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。 说明：代码与注释更新不同步，就像路网与导航软件更新不同步一样，如果导航软件严重滞后， 就失去了导航的意义。

## 谨慎注释掉代码。在上方详细说明，而不是简单地注释掉。如果无用，则删除。 说明：代码被注释掉有两种可能性：1）后续会恢复此段代码逻辑。2）永久不用。前者如果没 有备注信息，难以知晓注释动机。后者建议直接删掉（代码仓库保存了历史代码）。