前言

《杭州师范大学网络与信息安全实验室 JAVA 开发手册》是杭州师范大学信息安全实验室 Java 开发团队的经验总结，参考《阿里巴巴 JAVA 开发手册》泰山版的规约规范，以一位学生 Java 开发者的视角，给予网络与信息安全实验室的 Java 开发者们一些建议。

引用《阿里巴巴 JAVA 开发手册》中的一句话：“适当的规范和标准绝不是消灭代码内容的创造性、优雅性，而是限制过度个性化，以一种普遍认可的统一方式一起做事，提升协作效率，降低沟通成本。”因此，建议方向在 Java 开发的同学在阅读本手册的同时，搭配阿里巴巴 Java 开发规约 IDE 插件，在码代码的同时码出高效、码出质量。

目录

工程规范

（一）工程文档

1. 以文档驱动的开发相关文档分为：01_可行性研究报告、02_项目开发计划、03_需求规格说明书、04_概要设计说明书、05_详细设计说明书、06_用户操作手册、07_测试计划、08_测试分析报告、09_开发进度月报、10_项目开发总结报告、11_程序维护手册、12_软件问题报告、13_软件修改报告。

2．【推荐】不建议完全按照文档驱动来进行开发，但是建议 03_需求设计说明书、04_概要设计说明书必须完成。

3．【推荐】建议每一个项目前期经过需求评审再进行后续的设计和开发。需求评审可结合 Axure 原型图设计进行需求介绍。

4．【推荐】建议每一个项目前后端分离，接口文档可使用 RAP、showdoc 进行约定。

(二)应用规范

1．【推荐】项目 group 为网络与信息安全实验室缩写，artifactId 为项目名。分层可基本分为：

1.1 Controller 层：主要对访问控制进行转发，各类基本参数校验，或者不复用的业务简单处理。

1.2 Service 层：相对具体的的业务逻辑服务层。分为接口和 Impl 实现类。

1.3 Dao 层：数据访问层，与底层的 MySQL、Oracle、SQL SERVER 等进行数据交互。

1.4 外部接口或第三方平台：包括其他基础平台、其他公司的 HTTP 接口。

2．【参考】（分层异常处理规约）在 DAO 层，产生的异常类型有很多，无法用细粒度的异常进行 catch，使用 catch(Exception e)方式，并 throw new DAOException(e)，不需要打印日志，因为日志在 Service 层一定需要捕获并打印到日志文件中去，如果同台服务器再打日志，浪费性能和存储。在 Service 层出现异常时，必须记录出错日志到磁盘，尽可能带上参数信息，相当于保护案发现场。Web 层绝不应该继续往上抛异常，因为已经处于顶层，如果意识到这个异常将导致页面无法正常渲染，那么就应该直接跳转到友好错误页面，尽量加上友好的错误提示信息。开放接口层要将异常处理成错误码和错误信息方式返回。

3．【参考】分层领域模型规约：

3.1 DO（Data Object）：此对象与数据库表结构一一对应，通过 DAO 层向上传输数据源对象。

3.2 DTO（Data Transfer Object）：数据传输对象，Service 或 Manager 向外传输的对象。

3.3 BO（Business Object）：业务对象，可以由 Service 层输出的封装业务逻辑的对象。

3.4 Query：数据查询对象，各层接收上层的查询请求。注意超过 2 个参数的查询封装，禁止使用 Map 类来传输。

3.5 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。

MySQL 数据库

（一）建表规范

1. 【强制】表达是与否概念的字段，必须使用 is_xxx 的方式命名，数据类型是 unsigned tinyint （1 表示是，0 表示否）。
   说明：任何字段如果为非负数，必须是 unsigned。
   注意：POJO 类中的任何布尔类型的变量，都不要加 is 前缀，所以，需要在<resultMap>设置从 is_xxx 到 Xxx 的映射关系。数据库表示是与否的值，使用 tinyint 类型，坚持 is_xxx 的命名方式是为了明确其取值含 义与取值范围。
   正例：表达逻辑删除的字段名 is_deleted，1 表示删除，0 表示未删除。
2. 【强制】表名、字段名必须使用小写字母或数字，禁止出现数字开头，禁止两个下划线中间只 出现数字。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。
   说明：MySQL 在 Windows 下不区分大小写，但在 Linux 下默认是区分大小写。因此，数据库名、表名、 字段名，都不允许出现任何大写字母，避免节外生枝。
   正例：aliyun_admin，rdc_config，level3_name
   反例：AliyunAdmin，rdcConfig，level_3_name
3. 【强制】表名不使用复数名词。 说明：表名应该仅仅表示表里面的实体内容，不应该表示实体数量，对应于 DO 类名也是单数形式，符合表达习惯
4. 【强制】主键索引名为 pk_字段名；唯一索引名为 uk_字段名；普通索引名则为 idx_字段名。 说明：pk_ 即 primary key；uk_ 即 unique key；idx_ 即 index 的简称。
5. 【强制】小数类型为 decimal，禁止使用 float 和 double。 说明：在存储的时候，float 和 double 都存在精度损失的问题，很可能在比较值的时候，得到不正确的 结果。如果存储的数据范围超过 decimal 的范围，建议将数据拆成整数和小数并分开存储。
6. 【强制】如果存储的字符串长度几乎相等，使用 char 定长字符串类型。
7. 【强制】varchar 是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长度 大于此值，定义字段类型为 text，独立出来一张表，用主键来对应，避免影响其它字段索引效 率。
8. 【强制】表必备三字段：id, gmt_create, gmt_modified。 说明：其中 id 必为主键，类型为 bigint unsigned、单表时自增、步长为 1。gmt_create, gmt_modified 的类型均为 datetime 类型，前者现在时表示主动式创建，后者过去分词表示被动式更新。
9. 【推荐】库名与应用名称尽量一致。
10. 【推荐】如果修改字段含义或对字段表示的状态追加时，需要及时更新字段注释。

(二) 索引规范

1. 【强制】业务上具有唯一特性的字段，即使是组合字段，也必须建成唯一索引。 说明：不要以为唯一索引影响了 insert 速度，这个速度损耗可以忽略，但提高查找速度是明显的；另外， 即使在应用层做了非常完善的校验控制，只要没有唯一索引，根据墨菲定律，必然有脏数据产生。
2. 【强制】超过三个表禁止 join。需要 join 的字段，数据类型保持绝对一致；多表关联查

询时，保证被关联的字段需要有索引。 说明：即使双表 join 也要注意表索引、SQL 性能。

3. 【推荐】防止因字段类型不同造成的隐式转换，导致索引失效。
4. 【参考】创建索引时避免有如下极端误解：
   4.1：索引宁滥勿缺。认为一个查询就需要建一个索引。
   4.2：各啬索引的创建。认为索引会消耗空间、严重拖慢记录的更新以及行的新增速度。
   4.3：抵制惟一索引。认为惟一索引一律需要在应用层通过“先查后插”方式解决。

编程规范

(一)命名风格

1. 【强制】代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。 反例：_name / __name / $name / name_ / name$ / name__
2. 【强制】所有编程相关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，纯拼音命名方式更要避免采用。
   正例：ali / alibaba / taobao / cainiao/ aliyun/ youku / hangzhou 等国际通用的名称，可视同英文。
   反例：DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3
3. 【强制】类名使用 UpperCamelCase 风格，但以下情形例外：DO / BO / DTO / VO / AO / PO / UID 等。 正例：ForceCode / UserDO / HtmlDTO / XmlService / TcpUdpDeal / TaPromotion 反例：forcecode / UserDo / HTMLDto / XMLService / TCPUDPDeal / TAPromotion
4. 【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格。
   正例： localValue / getHttpMessage() / inputUserId
5. 【强制】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：MAX_STOCK_COUNT / CACHE_EXPIRED_TIME
   反例：MAX_COUNT / EXPIRED_TIME
6. 【强制】抽象类命名使用 Abstract 或 Base 开头；异常类命名使用 Exception 结尾；测试类 命名以它要测试的类的名称开始，以 Test 结尾。
7. 【强制】类型与中括号紧挨相连来表示数组。 正例：定义整形数组 int[] arrayDemo; 反例：在 main 参数中，使用 String args[]来定义。
8. 【强制】POJO 类中的任何布尔类型的变量，都不要加 is 前缀，否则部分框架解析会引起序列 化错误。
   说明：在本文 MySQL 规约中的建表约定第一条，表达是与否的值采用 is_xxx 的命名方式，所以，需要在 <resultMap>设置从 is_xxx 到 xxx 的映射关系。
   反例：定义为基本数据类型 Boolean isDeleted 的属性，它的方法也是 isDeleted()，框架在反向解析的时 候，“误以为”对应的属性名称是 deleted，导致属性获取不到，进而抛出异常。
9. 【强制】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.alibaba.ei.kunlun.aap.util、类名为 MessageUtils（此规则参考 spring 的 框架结构）

10. 【强制】杜绝完全不规范的缩写，避免望文不知义。
反例：AbstractClass“缩写”命名成 AbsClass；condition“缩写”命名成 condi，此类随意缩写严重降 低了代码的可阅读性。

11. 【推荐】为了达到代码自解释的目标，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。
正例：在 JDK 中，对某个对象引用的 volatile 字段进行原子更新的类名为：AtomicReferenceFieldUpdater。
反例：常见的方法内变量为 int a;的定义方式。

12. 【推荐】在常量与变量的命名时，表示类型的名词放在词尾，以提升辨识度。
正例：startTime / workQueue / nameList / TERMINATED_THREAD_COUNT
反例：startedAt / QueueOfWork / listName / COUNT_TERMINATED_THREAD

13. 接口和实现类的命名有两套规则：
1）【强制】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。
2）【推荐】如果是形容能力的接口名称，取对应的形容词为接口名（通常是–able 的形容词）。
正例：AbstractTranslator 实现 Translatable 接口。

14. 【参考】枚举类名带上 Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。
说明：枚举其实就是特殊的常量类，且构造方法被默认强制是私有。
正例：枚举名字为 ProcessStatusEnum 的成员名称：SUCCESS / UNKNOWN_REASON。

15. 【参考】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。
B) 领域模型命名规约
1） 数据对象：xxxDO，xxx 即为数据表名。
2） 数据传输对象：xxxDTO，xxx 为业务领域相关的名称。
3） 展示对象：xxxVO，xxx 一般为网页名称。
4） POJO 是 DO/DTO/BO/VO 的统称，禁止命名成 xxxPOJO。

（二）常量定义

1. 【强制】不允许任何魔法值（即未经预先定义的常量）直接出现在代码中。
反例： //本例中同学 A 定义了缓存的 key，然后缓存提取的同学 B 使用了 Id#taobao 来提取，少了下划线，导致故障。 String key = "Id#taobao_" + tradeId; cache.put(key, value);

2. 【强制】在 long 或者 Long 赋值时，数值后使用大写的 L，不能是小写的 l，小写容易跟数字 混淆，造成误解。 说明：Long a = 2l; 写的是数字的 21，还是 Long 型的 2。
3. 【推荐】如果变量值仅在一个固定范围内变化用 enum 类型来定义。 说明：如果存在名称之外的延伸属性应使用 enum 类型，下面正例中的数字就是延伸信息，表示一年中的第几个季节。

（三）代码格式

1. 【强制】如果是大括号内为空，则简洁地写成{}即可，大括号中间无需换行和空格；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。
2. 【强制】左小括号和右边相邻字符之间不出现空格；右小括号和左边相邻字符之间也不出现空格；而左大括号前需要加空格。详见第 5 条下方正例提示。
   反例：if (空格 a == b 空格)
3. 【强制】if/for/while/switch/do 等保留字与括号之间都必须加空格。
4. 【强制】任何二目、三目运算符的左右两边都需要加一个空格。
   说明：包括赋值运算符=、逻辑运算符&&、加减乘除符号等。
5. 【强制】注释的双斜线与注释内容之间有且仅有一个空格。
   正例： // 这是示例注释，请注意在双斜线之后有一个空格 String commentString = new String();
6. 【强制】在进行类型强制转换时，右括号与强制转换值之间不需要任何空格隔开。 正例：long first = 1000000000000L; int second = (int)first + 2;
7. 【强制】单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则：
   1）第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。
   2）运算符与下文一起换行。
   3）方法调用的点符号与下文一起换行。
   4）方法调用中的多个参数需要换行时，在逗号后进行。
   5）在括号前不要换行
8. 【强制】方法参数在定义和传入时，多个参数逗号后边必须加空格。

（四）OOP 规约

1. 【强制】避免通过一个类的对象引用访问此类的静态变量或静态方法，无谓增加编译器解析成本，直接用类名来访问即可。
2. 【强制】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。
3. 【强制】不能使用过时的类或方法。
   说明：java.net.URLDecoder 中的方法 decode(String encodeStr) 这个方法已经过时，应

该使用双参数 decode(String source, String encode)。接口提供方既然明确是过时接口，那么有义务同时提供新的接口； 作为调用方来说，有义务去考证过时方法的新实现是什么。

4. 【强制】Object 的 equals 方法容易抛空指针异常，应使用常量或确定有值的对象来调用 equals。
   正例：\"test\".equals(object);
   反例：object.equals(\"test\");
   说明：推荐使用 java.util.Objects#equals（JDK7 引入的工具类）。
5. 【强制】所有整型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 之间的赋值，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。
6. 【强制】定义数据对象 DO 类时，属性类型要与数据库字段类型相匹配。
7. 关于基本数据类型与包装数据类型的使用标准如下：
   1） 【强制】所有的 POJO 类属性必须使用包装数据类型。
   2） 【强制】RPC 方法的返回值和参数必须使用包装数据类型。
   3） 【推荐】所有的局部变量使用基本数据类型。

（五）集合处理

1. 【强制】关于 hashCode 和 equals 的处理，遵循如下规则：
   1） 只要重写 equals，就必须重写 hashCode。
   2） 因为 Set 存储的是不重复的对象，依据 hashCode 和 equals 进行判断，所以 Set 存储的对象必须重写 这两个方法。
   3） 如果自定义对象作为 Map 的键，那么必须覆写 hashCode 和 equals。
   说明：String 因为重写了 hashCode 和 equals 方法，所以我们可以愉快地使用 String 对象作为 key 来使 用。
2. 【强制】判断所有集合内部的元素是否为空，使用 isEmpty()方法，而不是 size()==0 的方式。
3. 【强制】使用 Map 的方法 keySet()/values()/entrySet()返回集合对象时，不可以对其进行添加元素操作，否则会抛出 UnsupportedOperationException 异常。
4. 【强制】在 subList 场景中，高度注意对父集合元素的增加或删除，均会导致子列表的遍历、增加、删除产生 ConcurrentModificationException 异常。
5. 【强制】使用集合转数组的方法，必须使用集合的 toArray(T[] array)，传入的是类型完全一 致、长度为 0 的空数组。
6. 【推荐】使用 entrySet 遍历 Map 类集合 KV，而不是 keySet 方式进行遍历。
   说明：keySet 其实是遍历了 2 次，一次是转为 Iterator 对象，另一次是从 hashMap 中取出 key 所对应的 value。而 entrySet 只是遍历了一次就把 key 和 value 都放到了 entry 中，效率更高。如果是 JDK8，使用 Map.forEach 方法。
   正例：values()返回的是 V 值集合，是一个 list 集合对象；keySet()返回的是 K 值集合，是一个 Set 集合对 象；entrySet()返回的是 K-V 值组合集合。
7. 【推荐】高度注意 Map 类集合 K/V 能不能存储 null 值的情况，如下：

Hashtable 不允许为 null 不允许为 null Dictionary 线程安全
ConcurrentHashMap 不允许为 null 不允许为 null AbstractMap 锁分段技术(JDK8:CAS)
TreeMap 不允许为 null 允许为 null AbstractMap 线程不安全
HashMap 允许为 null 允许为 null AbstractMap 线程不安全

（六）控制语句

1. 【强制】在一个 switch 块内，每个 case 要么通过 continue/break/return 等来终止，要么注释说明程序将继续执行到哪一个 case 为止；在一个 switch 块内，都必须包含一个 default
   语句并且放在最后，即使它什么代码也没有。说明：注意 break 是退出 switch 语句块，而 return 是退出方法体。
2. 【强制】当 switch 括号内的变量类型为 String 并且此变量为外部参数时，必须先进行 null 判断。
3. 【强制】在 if/else/for/while/do 语句中必须使用大括号。说明：即使只有一行代码，禁止不采用大括号的编码方式：if (condition) statements;
4. 【强制】三目运算符 condition? 表达式 1 : 表达式 2 中，高度注意表达式 1 和 2 在类型对齐 时，可能抛出因自动拆箱导致的 NPE 异常。
   说明：以下两种场景会触发类型对齐的拆箱操作： 1） 表达式 1 或表达式 2 的值只要有一个是原始类型。 2） 表达式 1 或表达式 2 的值的类型不一致，会强制拆箱升级成表示范围更大的那个类型。
5. 【推荐】表达异常的分支时，少用 if-else 方式，这种方式可以改写成：
   ```
   if (condition) {
    ...
    return obj;
    }
    // 接着写 else 的业务逻辑代码;
   ```
6. 【推荐】不要在其它表达式（尤其是条件表达式）中，插入赋值语句。说明：赋值点类似于人体的穴位，对于代码的理解至关重要，所以赋值语句需要清晰地单独成为一行。
   反例：
   ```
   public Lock getLock(boolean fair) {      // 算术表达式中出现赋值操作，容易忽略 count 值已经被改变
   threshold = (count = Integer.MAX_VALUE) - 1;      // 条件表达式中出现赋值操作，容易误认为是 sync==fair
   return (sync = fair) ? new FairSync() : new NonfairSync();
   }
   ```
7. 【推荐】循环体中的语句要考量性能，以下操作尽量移至循环体外处理，如定义对象、变量、 获取数据库连接，进行不必要的 try-catch 操作（这个 try-catch 是否可以移至循环体外）。

（七）注释规约

1. 【强制】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。

2. 【强制】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。
3. 【强制】所有的类都必须添加创建者和创建日期。
   正例:
   ```
   /**
   * @author yangguanbao
   * @date 2016/10/31
   */
   ```
4. 【强制】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。
5. 【强制】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。
6. 【推荐】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。
7. 【推荐】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。
8. 【推荐】在类中删除未使用的任何字段和方法; 在方法中删除未使用的任何参数声明与内部变 量。
9. 【参考】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。后者建议直接删掉即可, 假如需要查阅历史代码, 登录代码仓库即可。
10. 【参考】对于注释的要求: 第一、能够准确反映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 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。
11. 【参考】好的命名、代码结构是自解释的, 注释力求精简准确、表达到位。避免出现注释的一 个极端: 过多过滥的注释, 代码的逻辑一旦修改, 修改注释是相当大的负担。 反例:
    // put elephant into fridge put(elephant, fridge); 方法名 put, 加上两个有意义的变量名 elephant 和 fridge, 已经说明了这是在干什么, 语义清晰的代码不 需要额外的注释。

异常日志

(一) 异常处理

1. 【强制】Java 类库中定义的可以通过预检查方式规避的 RuntimeException 异常不应该通过 catch 的方式来处理, 比如: NullPointerException, IndexOutOfBoundsException 等

等。
说明：无法通过预检查的异常除外，比如，在解析字符串形式的数字时，可能存在数字格式错误，不得不 通过 catch NumberFormatException 来实现。 正例：if (obj != null) {...} 反例：try { obj.method(); } catch (NullPointerException e) {…}

2. 【强制】异常不要用来做流程控制，条件控制。 说明：异常设计的初衷是解决程序运行中的各种意外情况，且异常的处理效率比条件判断方式要低很多。
3. 【强制】catch 时请分清稳定代码和非稳定代码，稳定代码指的是无论如何不会出错的代码。 对于非稳定代码的 catch 尽可能进行区分异常类型，再做对应的异常处理。 说明：对大段代码进行 try-catch，使程序无法根据不同的异常做出正确的应激反应，也不利于定位问题， 这是一种不负责任的表现。 正例：用户注册的场景中，如果用户输入非法字符，或用户名称已存在，或用户输入密码过于简单，在程 序上作出分门别类的判断，并提示给用户。
4. 【强制】捕获异常是为了处理它，不要捕获了却什么都不处理而抛弃之，如果不想处理它，请 将该异常抛给它的调用者。最外层的业务使用者，必须处理异常，将其转化为用户可以理解的 内容。
5. 【强制】事务场景中，抛出异常被 catch 后，如果需要回滚，一定要注意手动回滚事务。
6. 【强制】finally 块必须对资源对象、流对象进行关闭，有异常也要做 try-catch。 说明：如果 JDK7 及以上，可以使用 try-with-resources 方式。
7. 【强制】不要在 finally 块中使用 return。 说明：try 块中的 return 语句执行成功后，并不马上返回，而是继续执行 finally 块中的语句，如果此处存 在 return 语句，则在此直接返回，无情丢弃掉 try 块中的返回点。
8. 【推荐】方法的返回值可以为 null，不强制返回空集合，或者空对象等，必须添加注释充分说 明什么情况下会返回 null 值。 说明：本手册明确防止 NPE 是调用者的责任。即使被调用方法返回空集合或者空对象，对调用者来说，也 并非高枕无忧，必须考虑到远程调用失败、序列化失败、运行时异常等场景返回 null 的情况。
9. 【推荐】防止 NPE，是程序员的基本修养，注意 NPE 产生的场景：
   1）返回类型为基本数据类型，return 包装数据类型的对象时，自动拆箱有可能产生 NPE。反例：public int f() { return Integer 对象}， 如果为 null，自动解箱抛 NPE。
   2） 数据库的查询结果可能为 null。
   3） 集合里的元素即使 isNotEmpty，取出的数据元素也可能为 null。
   4） 远程调用返回对象时，一律要求进行空指针判断，防止 NPE。
   5） 对于 Session 中获取的数据，建议进行 NPE 检查，避免空指针。
   6） 级联调用 obj.getA().getB().getC()；一连串调用，易产生 NPE。
   正例：使用 JDK8 的 Optional 类来防止 NPE 问题。
10. 【推荐】定义时区分 unchecked / checked 异常，避免直接抛出 new RuntimeException()，更不允许抛出 Exception 或者 Throwable，应使用有业务含义的自定义异常。推荐业界已定 义过的自定义异常，如：DAOException / ServiceException 等。

（二）日志规约

1. 【强制】应用中不可直接使用日志系统（Log4j、Logback）中的 API，而应依赖使用日志框架
（SLF4J、JCL--Jakarta Commons Logging）中的 API，使用门面模式的日志框架，有利于

维护和 各个类的日志处理方式统一。 说明：日志框架（SLF4J、JCL--Jakarta Commons Logging）的使用方式（推荐使用 SLF4J） 使用 SLF4J： import org.slf4j.Logger; import org.slf4j.LoggerFactory; private static final Logger logger = LoggerFactory.getLogger(Test.class); 使 用 JCL ： import org.apache.commons.logging.Log; import org.apache.commons.logging.LogFactory; private static final Log log = LogFactory.getLog(Test.class);

2. 【强制】所有日志文件至少保存 15 天，因为有些异常具备以“周”为频次发生的特点。对于 当天日志，以“应用名.log”来保存，保存在/home/admin/应用名/logs/目录下，过往日志格式为: {logname}.log.{保存日期}，日期格式：yyyy-MM-dd
   说明: 以 mppserver 应用为例，日志保存在/home/admin/mppserver/logs/mppserver.log，历史日志 名称为 mppserver.log.2016-08-01
3. 【强制】在日志输出时，字符串变量之间的拼接使用占位符的方式。
   说明：因为 String 字符串的拼接会使用 StringBuilder 的 append()方式，有一定的性能损耗。使用占位符仅 是替换动作，可以有效提升性能。
   正例：logger.debug("Processing trade with id: {} and symbol: {}", id, symbol).
4. 【强制】生产环境禁止直接使用 System.out 或 System.err 输出日志或使用 e.printStackTrace()打印异常堆栈。
   说明：标准日志输出与标准错误输出文件每次 Jboss 重启时才滚动，如果大量输出送往这两个文件，容易 造成文件大小超过操作系统大小限制。
5. 【强制】异常信息应该包括两类信息：案发现场信息和异常堆栈信息。如果不处理，那么通过 关键字 throws 往上抛出。 正例：logger.error(各类参数或者对象 toString() + "_" + e.getMessage(), e);
6. 【强制】日志打印时禁止直接用 JSON 工具将对象转换成 String。
   说明：如果对象里某些 get 方法被重写，存在抛出异常的情况，则可能会因为打印日志而影响正常业务流 程的执行。
   正例：打印日志时仅打印出业务相关属性值或者调用其对象的 toString()方法。
7. 【推荐】可以使用 warn 日志级别来记录用户输入参数错误的情况，避免用户投诉时，无所适从。如非必要，请不要在此场景打出 error 级别，避免频繁报警。
   说明: 注意日志输出的级别，error 级别只记录系统逻辑出错、异常或者重要的错误信息。
8. 【推荐】尽量用英文来描述日志错误信息，如果日志中的错误信息用英文描述不清楚的话使用 中文描述即可，否则容易产生歧义

安全规约

1. 【强制】隶属于用户个人的页面或者功能必须进行权限控制校验。 说明：防止没有做水平权限校验就可随意访问、修改、删除别人的数据，比如查看他人的私信内容。
2. 【强制】用户敏感数据禁止直接展示，必须对展示数据进行脱敏。 说明：中国大陆个人手机号码显示为:137****0969，隐藏中间 4 位，防止隐私泄露。
3. 【强制】用户输入的 SQL 参数严格使用参数绑定或者 METADATA 字段值限定，防止 SQL 注入， 禁止字符串拼接 SQL 访问数据库。
   反例：某系统签名大量被恶意修改，即是因为对于危险字符 # --没有进行转义，导致数据库更新时，where 后边的信息被注释掉，对全库进行更新。
4. 【强制】用户请求传入的任何参数必须做有效性验证。
   说明：忽略参数校验可能导致：

1) page size 过大导致内存溢出
2) 恶意 order by 导致数据库慢查询
3) 缓存击穿
4) SSRF
5) 任意重定向
6) SQL 注入，Shell 注入，反序列化注入
7) 正则输入源串拒绝服务 ReDoS
Java 代码用正则来验证客户端的输入，有些正则写法验证普通用户输入没有问题，但是如果攻击人员使用 的是特殊构造的字符串来验证，有可能导致死循环的结果。

5. 【强制】禁止向 HTML 页面输出未经安全过滤或未正确转义的用户数据。
6. 【强制】表单、AJAX 提交必须执行 CSRF 安全验证。说明：CSRF(Cross-site request forgery)跨站请求伪造是一类常见编程漏洞。对于存在 CSRF 漏洞的应用/ 网站，攻击者可以事先构造好 URL，只要受害者用户一访问，后台便在用户不知情的情况下对数据库中用 户参数进行相应修改。
7. 【强制】URL 外部重定向传入的目标地址必须执行白名单过滤。
8. 【强制】在使用平台资源，譬如短信、邮件、电话、下单、支付，必须实现正确的防重放的机制，如数量限制、疲劳度控制、验证码校验，避免被滥刷而导致资损。
说明：如注册时发送验证码到手机，如果没有限制次数和频率，那么可以利用此功能骚扰到其它用户，并 造成短信平台资源浪费。
9. 【推荐】发贴、评论、发送即时消息等用户生成内容的场景必须实现防刷、文本内容违禁词过 滤等风控策略。

IDEA 插件推荐

1. Alibaba Java Coding Guidelines

   为了让开发者更加方便、快速将规范推动并实行起来，阿里巴巴基于手册内容，研发了一套自动化的 IDE 检测插件（IDEA、Eclipse）。该插件在扫描代码后，将不符合规约的代码按 Blocker/Critical/Major 三个等级显示在下方，甚至在 IDEA 上，我们还基于 Inspection 机制提供了实时检测功能，编写代码的同时也能快速发现问题所在。对于历史代码，部分规则实现了批量一键修复的功能，如此爽心悦目的功能是不是很值得拥有？提升代码质量，提高团队研发效能，插件将会一路同行。
2. JRebel + XRebel
   JRebel 为热部署插件，免去重启。支持 run 模式和 debug 模式的热部署启动。
   XRebel 是不间断运行在 web 应用的交互式分析器，当发现问题会在浏览器中显示警告信息。XRebel 会实时监测应用代码的性能指标和可能会发生的问题。
3. Free MyBatis plugin
   free-idea-mybatis 是一款增强 idea 对 mybatis 支持的插件，主要功能如下：
   生成 mapper xml 文件
   快速从代码跳转到 mapper 及从 mapper 返回代码
   mybatis 自动补全及语法错误提示
   集成 mybatis generator gui 界面
4. Maven Helper
   主要功能如下：查找和排除冲突依赖项的简便方法，为包含当前文件或根模块的模块运

行/调试 maven 目标的操作，运行/调试当前测试文件的操作

5. RestfulToolkit

   一套 RESTful 服务开发辅助工具集。

   1) 根据 URL 直接跳转到对应的方法定义（Ctrl \ or Ctrl Alt N）;

   2) 提供了一个 Services tree 的显示窗口;

   3) 一个简单的 http 请求工具;

   4) 在请求方法上添加了有用功能: 复制生成 URL;,复制方法参数...

   5) 其他功能: java 类上添加 Convert to JSON 功能，格式化 json 数据（Windows: Ctrl + Enter; Mac: Command + Enter）。

6. Lombok

   Lombok 能以简单的注解形式来简化 java 代码，提高开发人员的开发效率。例如开发中经常需要写的 javabean，都需要花时间去添加相应的 getter/setter，也许还要去写构造器、equals 等方法，而且需要维护，当属性多时会出现大量的 getter/setter 方法，这些显得很冗长也没有太多技术含量，一旦修改属性，就容易出现忘记修改对应方法的失误。Lombok 能通过注解的方式，在编译时自动为属性生成构造器、getter/setter、equals、hashcode、toString 方法。