支付宝

编码规范

简化版

基本信息及修订记录				
创建时间：2010-05-24				
版本	**时间**	**作者**	**修订章节**	**修订内容**
1.0	2010-06-01	SQPG	全部	简版编码规范

目 录

支付宝代码规范（简化版）

一． 操作规范

1.模板及格式化

支付宝的开发人员必须保证代码格式化的一致性，否则可能会导致代码冲突，轻微的耗费人力合并代码；严重时可能导致代码丢失，引起 bug 或者故障。

- 开发人员必须配置 ALIPAY 的 codetemplates.xml 代码模板文件。
- 开发人员必须配置 ALIPAY 的 AlipayFormatter.xml 代码格式化文件。
- 每次提交代码之前，必须对 java 代码 format。

最新模板文件的下载地址：

```
http://home.alipay.net/downloads
```

Eclipse 中配置的位置：

```
Window->Preferences->Java->Code templates
Window->Preferences->Java->Formatter
```

2.代码提交

- 为防止冲突，任何时候，代码（及配置文件）提交前，先从 CC 或者 SVN 中更新代码和配置文件，以及早发现不兼容的代码变更和冲突。
- 提交代码（及配置文件）时，如果发生冲突时，先看历史说明，再找相关人员确认，坚决不允许强制覆盖。
- 每次提交代码之前，必须检查是否有 eclipse warning，并 FIX 所有的 warning（由 dalgen 等自动生成、不允许人工修改的代码例外）。

- 开发过程定期使用 FindBugs 扫描代码，合并代码时不允许出现高等级问题。

3. 垃圾清理

- 对于从来没有用到的或者被注释的方法，变量，类，配置文件，动态配置属性等要坚决从系统中清理出去，避免造成过多垃圾。

二. 日志规范

4. 日志输出

- 生产代码禁止以 System 及 Throwable.printStackTrace 的方式输出日志信息，必须用 Logger 替代。
- 对 trace/debug/info 级别的日志输出，必须使用条件输出形式，否则大量的日志会增加来自对象 toString 的性能成本。
- 对于日志的打印，任何情况下都不允许日志错误导致业务失败。
- 对于异常堆栈的输出，必须以 log.XXX（“msg”,e)的形式输出，禁止 log.XXX（“msg”+e)的错误形式。

5. 错误日志

- 所有错误日志必须输出到 error.log（webx 系统）或 common-error.log（sofa 系统）中。目前线上所有的错误日志会自动生成报告，限期安排人员分析与解决。
- 对于由于系统原因造成业务处理失败的事件，需要记录错误日志。非系

统原因的业务处理失败，不应该记录错误日志（推荐使用 warn 级别），避免错误日志过大，影响紧急情况下的故障分析与诊断。

三． 注释规范

6.基本原则

- 对于一个完整的类，应该包括 ALIPAY 的版权注释、类的说明注释、类成员变量注释，以及 public、protected、private 方法的注释（setter、getter、接口的实现方法除外），具体例子可以参考 Money 类。
- 必须保证代码和注释的一致性。
- 正确区分使用文档注释和实现逻辑注释。

四． 安全规范

7.敏感信息的保护

- 用户的敏感信息包括密码、短信验证码、支付验证码、身份证号、银行卡号、银行密钥，商户密钥等信息；用户敏感信息不能泄露，否则可能会带来不安全因素。可能会导致敏感信息泄露的方式有：Logger、URL 的 get 参数（因为 URL 的 get 参数会在 apache 日志中被输出）。

8.WEB 安全

- 对于前台的 web 页面，必须加上表单防重复提交功能；对于业务关键字

段需要加上防篡改的功能，否则可能造成业务重复执行或者被客户端恶意修改。

- 对于前台的 web 页面，严禁出现方便调试的后门页面。
- 不允许出现页面向自身重定向、或者多个页面间相互重定向的情况。如果控制不当，很容易造成系统宕机。

五． 通用规范

9.金额的使用

资金计算与处理必须使用 Money 类，禁止使用浮点数（Double，Float 等）直接进行处理，否则会有精度问题。

10． 枚举的使用

对于枚举，必须使用 JDK1.5 自带的 Enum，不允许新建阿里枚举，否则可能会有 Xfire 调用、死锁等问题。

11． URL 使用

- 外部重定向地址必须使用 URIBroker 生成，否则由于线上采用的是 https 协议，会导致浏览器提示“不安全提示”。

12． 配置信息的使用

- 避免将 UserID、AccountNO、URL、文件名、系统开关参数、业务规

则的可变参数等硬编码。

13. 异常处理

- 捕捉到的异常，不允许不做任何处理就截断，至少要记入日志，或重新抛出。
- 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。

14. 资源的使用

对系统资源的访问，使用后必须释放系统资源。这类资源包括：文件流、线程、网络连接、数据库连接等。

- 对于文件、流的 `IO` 操作，必须通过 `finally` 关闭。
- 对于线程，线程资源必须通过线程池提供，不允许在应用中自行显式创建线程。
- 对于网络连接与数据库连接，必须由框架通过连接池提供，不允许应用中自行建立网络与数据库连接。

15. 本地事务操作

- 对于业务逻辑上不允许并发访问的数据（例如具有全局唯一性的数据，涉及到总和类的数据等），必须采用事务和加锁的方式进行处理。
- 对于业务逻辑上要求数据完整性的数据（例如同时操作多个表，对同一个表反复进行操作等），必须采用事务的方式进行处理。

16. 线程安全处理

虽然容器会负责多线程的处理，但是程序中还是会遇到很多线程安全的问题，开发人员必须注意并发处理，否则可能导致死锁或者资损。

- 线程上下文变量的设置与清除必须配对。
- 静态 `Util` 或单例必须是线程安全的。
- `DateFormat` 类是非线程安全的，类变量使用时会被破坏。每次使用都要重新构造，或者使用 `DateUtil` 工具类。
- 为记录加锁时，需要保持一致的加锁顺序，否则可能会造成死锁。