开发规范

更新日期	版本号	修订类型	作者
2017/06/10	1.0.0	根据信息技术部要求制定开发规范	张羽
2017/06/14	1.0.1	补充：五、应用规范	张羽
2017/06/18	1.0.2	补充：五、(四)、其他规范	张羽

提高部门软件项目开发的效率和质量，要求所有开发人员必须遵守。

开发规范在项目开发过程中具有非常重要的指导意义：

- 提高项目开发的整体质量；
- 提高代码的可读性；

制定开发规范的目标和要求：

- 统一编码风格；
- 统一命名规范；
- 统一项目结构；

开发规范根据当前的实际情况进行制订，保证了可操作性，在后续的代码审查过程中发现的问题会补充进来。

为了配合开发规范的实施，平台会提供相应的辅助工具：格式化模板、注释模板、代码块模板、JSHint 插件、FindBugs 插件、测试覆盖率插件。

一、 总则

★ 【强制】程序文件要求必须以 UTF-8 进行编码，不包含签名（EmEditor 的编码中有此选项）；

★ 【强制】所有编程相关的命名使用英文或者英文缩写，严禁使用拼音或拼音与英文混合的方式，更不允许直接使用中文汉字的方式。

★ 【强制】程序源码缩进必须使用空格键，不允许使用 TAB 键。

★ 【强制】java/js 代码必须经 Ctrl+Shift+F 进行格式化。

★ 【强制】IDE 中的代码警告必须合理清除。

★ 【强制】findbugs 扫描出来的问题必须都解决掉。

★ 【强制】开发人员必须代码自测覆盖率 100%才能提交。

二、 数据库规范

(一) 格式规范

★ 数据库脚本文件(*.sql)的文件格式必须为纯文本，不允许使用包含格式文本的文件格式（比如 word 文档）；

★ 代码格式（大小写、缩进、换行等）以 PL/SQL Developer 导出的格式为准（或者[编辑]/]PLSQL 美化器]处理后的结果为准）。

(二) 命名规范

【表空间】TS_表空间名称

【表命名】系统简称_[模块简称_]具体名称

【字段】没有前缀，多个单词用下划线分割

【主键】PK_表名

【索引】IDX_表名_字段名

【唯一所有】UK_表名_字段名

【序列】SEQ_表的名称

（三） 创建表

★ 表空间

严禁用系统表空间作为用户默认表空间；严禁在系统表空间上创建用户数据库对象；

严禁在 SYSTEM/SYS 等系统用户下，创建用户数据库对象；

创建表时数据和索引建议在不同的表空间。

提示：通过 PL/SQLDeveloper 导出 sql 后，删除语句中的表空间相关内容。

★ 必须的字段

ID NUMBER -- 主键，一般通过序列进行取值

CREATE_TIME TIMESTAMP -- 创建时间，插入数据时的应用服务器时间

MODIFY_TIME TIMESTAMP -- 最后修改时间，每次记录修改都要更新为应用服务器的当前时间

VALIDATE_STATE -- 有效状态（1-有效、0-无效，逻辑删除用）

（四） SQL 规范

★ SQL 脚本中不允许出现“*”，必须用实际的字段名代替，INSERT、SELECT 语句必须指定要操作的字段名

★ SQL 语句的 WHERE 子句中应尽可能将字段放在等式左边，将计算操作放在等式的右边，否则将不会使用索引。

示例：

错误的用法： WHERE to_char(birthday,'yyyy-mm-dd') > '1990-01-01'

正确的用法： WHERE birthday > to_date('1990-01-01', 'yyyy-mm-dd')

（五） 数据字典

同一意义的字段，在系统内、系统间必须保持字段名称、类型、长度的一致。

约定数据字典以供参考：

名称	字段名称	类型及长度	备注
身份证号			

合同号			
客户号			
银行卡号			
手机号			

三、 java 规范

(一) 基本规范

【强制】开发前，必须首先把 IDE 的 Preferences/General/Workspace 设置中的 text file encoding 设置为 UTF-8，文件的换行符设置为 Unix 格式，不要使用 windows 格式。

【强制】java 包命名全部小写。类名使用 UpperCamelCase 风格（首字母大写，不同单词首字母大写），DTO、DAO 等后缀例外。

【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格（首字母小写，不同单词首字母大写）。

【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。

【强制】POJO 类中的任何布尔类型（boolean）的变量，都不要加 is，否则部分框架解析会引起序列化错误。

【强制】代码块缩进 4 个空格，如果使用 tab 缩进，请设置成 1 个 tab 为 4 个空格。

【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；任何运算符左右必须加一个空格——可以通过 Ctrl+Shift+F 进行格式化。

【强制】不允许使用过时的（@Deprecated）类和方法

(二) 包命名规范

★ 包命名规范：com.jy.系统名[.子系统名|模块名].功能名

其中：com.jy.是固定的包名，系统名为项目名。

比如：开发平台名称为 platform，那么：com.jy.platform.为开发平台的基础包名。

贷前系统名称为 loanb，那么：com.jy.loanb.为贷前系统的基础包名。

包名命名原则是每个项目的基础包名都不一样，从而保证整个捷越的所有软件系统模块都有独立命名空间，保证模块拆分和合并不会造成代码冲突。

★ 模块下各个包命名规范：

dto 包：放和数据库表对应的 POJO；

dao 包：放 mybaits 的 Mapping 文件和 DAO 接口类；

controller 包：放 spring mvc 的 controller 层代码；

rest 包：放 restful 服务类；

service 包：放实现业务逻辑的 service 类，以及 api 包中接口的实现类；

api 包：放为 RPC 暴露的接口；

（三） 类名的命名规范

类名要用名词或名词短语定义；

不同功能类别下类的命名规则如下（SSM 框架内的代码，基本都可以通过工具生成）：

Controller：业务名称+ Controller；

Rest：业务名称+ Rest；

Api：功能+Api；

Service：功能+Service；

dto：表名+DTO；

dao：表名+DAO；

Filter：业务名称+Filter；

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。

（四） 注释规范

★ 文件注释：在每个文件的头部都应该包含该文件的功能、作用、作者（必须是中文姓名）、日期以及创建、修改记录等。并在其中使用 SVN 标记自动跟踪版本变化及修改记录等信息。注意是/**/注释而不是/***/JavaDoc 注释。

★ 类注释：在类、接口定义之前当对其进行 javaDoc 注释，包括类、接口的目的、作用、功能、继承于何种父类，实现的接口、实现的算法、使用方法、示例程序等，在作者和版本域中使用 SVN 标记自动跟踪版本变化等，具体参看注释模板。

★ 方法注释：依据标准 JavaDoc 规范对方法进行注释，以明确该方法功能、作用、各参数含义以及返回值等。参数注释时当注明其取值范围等；返回值当注释出失败、错误、异常时的返回情况，什么情况返回 null；异常当注释出什么情况、什么时候、什么条件下会引发什么样的异常。

（五） 异常及日志规范

★【强制】异常不要用来做流程控制，条件控制，因为异常的处理效率比条件分支低。

★【强制】严禁对大段代码进行 try-catch。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。

★【强制】严禁捕获异常却什么都不处理就又抛出，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。

★ 捕获（catch）异常，如果处理后还继续向上抛出，不用记录 ERROR 日志、不用打印堆栈信息；如果捕获异常后处理掉，必须在 catch 语句块中记录 ERROR 级别的日志。

★ 在 finally 代码块中释放资源时 catch 的异常？？？

★【强制】不能在 finally 块中使用 return，finally 块中的 return 返回后方法结束执行，不会再执行 try 块中的 return 语句。

★【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 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。

★【强制】对 trace/debug/info 级别的日志输出，必须使用条件输出形式或者使用占位符的方式。

★【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。

★ 日志级别规范：

error：系统异常（catch 住的）

warn：系统控制住的人为以外处理，比如用户输入参数错误。

info：业务信息

debug：调试信息，生产环境正常后严禁输出

（六） 数据库事务规范

★ 系统的数据库事务控制在 Service 层，统一通过 Spring 的 AOP 进行配置管理，所有数据库的增删改操作必须进行统一的事务控制。

★ 新增操作方法名必须以 insert 开头；更新操作方法名必须以 update 开头；删除操作方法名必须以 delete 开头；查询单个对象的操作方法必须以 get 开头；查询数据列表的操作方法必须以 query 或 select 开头，或者叫 get****List。

（七） Rest 规范

Rest 类上必须使用注解@Controller

Rest 服务路径规范（Rest 类上的注解@RequestMapping）：/api/子系统/模块名/类名

方法上的注解@RequestMapping：/方法名/版本号

（八） 技术规范

★ 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 方法。推荐使用 java.util.Objects 的 equals 方法（JDK7 引入的工具类）

★ 关于基本数据类型与包装数据类型的使用标准如下：

1） 所有的 POJO 类属性必须使用包装数据类型。

2） RPC 方法的返回值和参数必须使用包装数据类型。

3） 所有的局部变量推荐使用基本数据类型。

★ 序列化类新增属性时，请不要修改 serialVersionUID 字段，避免反序列失败；如果完全不兼容升级，避免反序列化混乱，那么请修改 serialVersionUID 值。

★ 构造方法里面禁止加入任何业务逻辑，如果有初始化逻辑，请放在 init 方法中。

★ 使用索引访问用 String 的 split 方法得到的数组时，需做最后一个分隔符后有无内容的检查，否则会有抛 IndexOutOfBoundsException 的风险。

★ Map/Set 的 key 为自定义对象时，必须重写 hashCode 和 equals。

★ 通过 foreach 循环进行集合遍历时不要进行元素的 remove/add 操作。remove 元素请使用 Iterator 方式，如果并发操作，需要对 Iterator 对象加锁。

★ 使用 entrySet 遍历 Map 类集合 KV，而不是 keySet 方式进行遍历，前者效率高。

★ 利用 Set 元素唯一的特性，可以快速对另一个集合进行去重操作，避免使用 List 的 contains 方法进行遍历去重操作。

★ SimpleDateFormat 是线程不安全的类，一般不要定义为 static 全局变量，方法内的局部变量没问题。

★ 并发修改同一记录时，避免更新丢失，要么在应用层加锁，要么在数据库层使用乐观锁，使用 version 作为更新依据。

★ 对于“明确停止使用的代码和配置”，如方法、变量、类、配置文件、动态配置属性等要坚决从程序中清理出去，避免造成过多垃圾。

★ xml 配置中参数注意使用：#{}，#param# 不要使用${} 此种方式容易出现 SQL 注入。

★ 不要写一个大而全的数据更新接口，传入为 POJO 类，不管是不是自己的目标更新字段，都进行 update table set c1=value1,c2=value2,c3=value3; 这是不对的。执行 SQL 时，尽量不要更新无改动的字段，一是易出错；二是效率低；三是数据库的 binlog 增加存储。

★ 所有的对象（包括基本数据类型的包装类对象）之间值的比较，全部使用 equals 方法比较，不允许使用==比较。

★ 集合初始化时，尽量指定集合初始值大小，以避免扩容影响效率。

★ 工程直接依赖的 jar 包，以及间接依赖的 jar 包，都必须在 Maven 的 pom.xml 文件中配置，以避免发生版本不一致的问题。

四、 前端规范

★ js 代码必须经过 Ctrl+Shift+F 快捷键进行格式化；

★ IDE 检查出来的警告信息必须合理解决掉；

★ js 代码必须经过 JSHint 工具的静态检查（培训方法：提供一个很烂的 js 代码，让新员工根据工具去优化||可以安装 jshint eclipse 插件）；

★ js 中变量、函数、常量的命名遵循 java 的命名规范；

★ 除特殊情况，需把 js 块放在页面在 head 位置外，应尽量放到</body>（页面最后）之前；

★ 在语句结束后一定要加入“;”进行结束。如果放在单行中，不加放“;”系统会报错；

★ 在定义变量时应加入 var，由其在局部变量中，如果不加入 var 来定义变量，变量奖是全局变量，在方法执行完成后，不会释放内存，同时也容易于全局变量混淆；

五、 应用规范

（一） 应用部署及配置规范

★ 应用以目录形式统一部署到/home/jyapp 目录下，在 tomcat 的 server.xml 配置文件中通过docBase指定目录,严禁以 war包的形式放到发布到 tomcat 的webapps下。

（二） 应用升级规范

★ 应用升级操作步骤必须严格按照以下流程

1、首先必须全量备份应用目录，格式 xxxx.yyyyMMddHHmm.tgz，严禁只备份单个文件，这样便于回退，备份目录 xxxx.bak。

2、按升级操作说明进行升级操作、验证。

（三） 数据库操作规范

★ 在生产环境，严禁使用在 PL/SQL Developer 中执行 SELECT * FROM TABLE_NAME FOR UPDATE 的方式更新数据，这样将会锁表影响业务；如必须更新，直接使用 UPDATE 语句进行。

（四） 其他规范

★ 在生产环境配置变更（特别是性能调优类的），需要进行灰度发布，只变更一部分，经变更前后对比，能明确效果再全部更新，无效果则回退，避免毫无意义的“优化”。