恒大智慧小区项目

**JAVA编码规范**

**恒大智慧小区项目团队**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **文档信息** | | | | |
| 项目名称 | 恒大智慧小区项目 | | | |
| 文档名称 | JAVA编码规范 | | | |
| 项目编号 | G3P1T3M1 | | 文档编号 |  |
| 作者 | 程立强 | | 文件类型 | 交付件 |
| 审批者 |  | | 审批日期 |  |
| 文件名称 | EGSC\_公共平台\_JAVA编码规范\_v1.0 | | | |
| 内容范围 | JAVA编码规范 | | | |
| 适用对象 | 设计人员、开发人员、测试人员 | | | |
| **版本信息** | | | | |
| 版本编号 | 版本日期 | 作者 | 修改描述 | |
| v0.1 | 2017-11-14 | 程立强 | 初稿 | |
| V1.0 | 2017-12-08 | 林 钟 | 修订 | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |

目 录

1 引言 6

1.1 编写目的 6

1.2 背景 6

1.3 参考资料 6

1.4 术语定义及说明 6

2 编程规范 8

2.1 命名规范 8

2.1.1 工程命名 8

2.1.2 包命名 8

2.1.3 类命名 9

2.1.4 接口命名 9

2.1.5 常量命名 9

2.1.6 变量命名 9

2.1.7 方法命名 9

2.1.8 代码格式 9

2.2 编码约束 10

2.2.1 equals 10

2.2.2 避免循环中数据库操作 10

2.2.3 循环体字符串拼接 10

2.2.4 避免使用多线程 11

2.3 注释规范 11

2.3.1 文件头 11

2.3.2 类 11

2.3.3 接口 12

2.3.4 常量与成员变量 12

2.3.5 方法 12

2.3.6 代码块 13

3 日志规范 15

4 异常处理 16

4.1 抛出异常类型 16

4.2 避免异常捕获后什么也不做 16

4.3 避免循环内采用异常捕获 16

5 单元测试 17

6 安全规约 18

6.1 敏感数据脱敏 18

6.2 防重复提交 18

6.3 防sql注入 18

6.4 用户请求参数有效性验证 18

6.5 HTML页面输出 19

6.6 CSRF安全过滤 19

# 引言

## 编写目的

本详细设计说明书的目的是，规范应用系统的代码组织风格，改善软件的可读性，让开发人员尽快而彻底地理解新的代码。最大限度的提高团队开发的合作效率让，开发人员养成好的编码习惯，甚至锻炼出更加严谨的思维。减少软件在整个生命周期中的维护成本。

预期读者：开发人员、测试人员、项目经理。

## 背景

作为恒大集团进军互联网社区服务产业的旗舰公司，恒腾网络受集团指派，开展“智慧小区”的研究和建设工作。恒大希望通过“智慧小区”的建设，一方面通过给客户提供差异化的体验，实现对恒大目前产品的增值，另一方面，实现恒大基于智慧小区的服务能力升级。工程将分阶段进行。未来预期本项目成果将应用到恒大集团800多家小区，及其他房地产合作商。

本项目为“恒大智慧小区“工程的第一阶段，即通过快速而集中的开发建设，预计在2018年3月30日前对整体方案实现现场验证。

## 参考资料

## 术语定义及说明

|  |  |
| --- | --- |
| 术语 | 术语解释 |
|  |  |

# 编程规范

## 命名规范

采用英文单词，禁止拼音，中文

注：下文采用用户管理应用为例

### 工程命名

应用的英文名称（参见应用与模块文档），采用字母小写，后续单词用“\_”连接

示例：

公共框架： common\_framework

智能设备轻量级接入网关：smart\_device\_gateway

### 包命名

（小写）com.eg.egsc.scp.应用缩写.模块英文名称.层

示例：

角色管理：

com.eg.egsc.scp.usermgmt.rolemanagement.entity(对象）

com.eg.egsc.scp.usermgmt.rolemanagement.service(接口）

com.eg.egsc.scp.usermgmt.rolemanagement.service.impl（实现）

### 类命名

首字母大写驼峰命名（UpperCamelCase）

接口实现类：+Impl

### 接口命名

首字母大写驼峰命名（UpperCamelCase）

### 常量命名

大写，单词之间加下划线 “\_”

### 变量命名

首字母小写驼峰命名（lowerCamelCase）

### 方法命名

首字母小写驼峰命名（lowerCamelCase）

### 代码格式

采用开发机提供的Eclipse内置格式(参见Eclipse->Preference->Java->Code Style->Formatter)：

在第一次代码提交以前，先做全局format，后提交；

在后续修订提交时，只对新增或修改部分做format，后提交；

## 编码约束

### equals

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");

### 避免循环中数据库操作

避免在循环内部调用数据库操作，以防止出现性能问题

### 循环体字符串拼接

循环体内，字符串的连接方式，使用StringBuilder的append方法进行扩展。

说明：反编译出的字节码文件显示每次循环都会new出一个StringBuilder对象，然后进行

append操作，最后通过toString方法返回String对象，造成内存资源浪费。

反例：

String str = "start";

for (int i = 0; i < 100; i++) {

str = str + "hello";

}

### 避免使用多线程

避免使用多线程，如果实在不可避免，请提出说明，由架构评审后决定

## 注释规范

采用开发机eclipse带的Code Template（参见Eclipse->Preference->Java->Code Style->Code Template)

### 文件头

/\*\*

\* Copyright 2017 - 2025 Evergrande Group

\*/

### 类

需要有JavaDoc注释，说明类的用途

采用

/\*\*

\* **TODO**

\* **@Author** zhansan

\* **@Since** 2018年1月9日

\*/

### 接口

与类注释要求相同

### 常量与成员变量

需要在常量或成员变量上一行有说明

示例：

/\*\*

\* 用户id

\*/

private String userName;

### 方法

接口方法与公共方法(除get/set)必须有说明：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

示例：

/\*\*

\* **TODO**

\* **@param** userDto

\* **@return** int

\*/

### 代码块

方法内部的代码块注释，

单行采用//

多行采用：/\*

\*/

多步骤业务逻辑注释，要标明顺序步骤，以用户注销操作为例：

//1.查询用户状态(步骤1)

code

//2.判断当前人是否有管理权限(步骤2)

code

//3.执行账户注销(步骤3)

code

# 日志规范

SLF4J中的API，使用门面模式的日志框架，后端适配Log4j2，有利于维护和各个类的日志处理方式统一。

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

private static final Logger logger = LoggerFactory.getLogger(Abc.class);

# 异常处理

## 抛出异常类型

业务如果不能自己处理异常，则抛出一个RuntimeException类型异常，不要直接抛出Exception，除非强制要求调用方处理。

## 避免异常捕获后什么也不做

至少有日志信息输出

## 避免循环内采用异常捕获

# 单元测试

在提交测试前，先完成单元测试

# 安全规约

## 敏感数据脱敏

用户敏感数据应该避免直接展示，需要对展示数据进行脱敏。

## 防重复提交

例如：

下单，收费等，需要避免重复提交

登录，需要设置重试次数，避免恶意攻击

## 防sql注入

SQL参数严格使用参数绑定，避免字符串拼接的SQL访问数据库

## 用户请求参数有效性验证

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：

page size过大导致内存溢出

恶意order by导致数据库慢查询

任意重定向

反序列化注入

正则输入源串拒绝服务ReDoS

（注：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）

## HTML页面输出

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

## CSRF安全过滤

表单、AJAX提交必须执行CSRF安全过滤。

（注：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情况下对数据库中用户参数进行相应修改）