# **二手图书交易系统编码规范说明文档**

## **1. 目的**

为规范开发人员的代码编写，提高代码质量，特制定此编码规范。通过此规范可以：

· 增强代码的可读性和可维护性

· 降低程序出错的概率

· 提高开发效率

· 规范团队开发模式

· 便于新成员快速融入团队

· 降低项目维护成本

· 提高团队协作效率

## **2. 适用范围**

本规范适用于基于Spring Boot的二手图书交易系统开发，涵盖：

· 后端Java代码编写

· 数据库设计和访问

· 接口设计和实现

· 配置文件管理

· 工具类开发

· 文档编写

所有开发人员必须严格遵守本规范。

## **3. 项目结构规范**

### **3.1 包结构规范**

**src/main/java/com/**

**├── annotation/     // 自定义注解**

**├── IgnoreAuth.java     // 忽略登录校验注解**

**└── LoginUser.java      // 登录用户注解**

**├── config/         // 配置类**

**├── AlipayConfig.java   // 支付宝配置**

**├── InterceptorConfig.java  // 拦截器配置**

**└── MybatisPlusConfig.java  // MyBatis-Plus配置**

**├── controller/     // 控制器层**

**├── AddressController.java  // 地址管理**

**├── OrderController.java    // 订单管理**

**└── UserController.java     // 用户管理**

**├── dao/           // 数据访问层**

**├── AddressDao.java        // 地址数据访问**

**├── OrderDao.java          // 订单数据访问**

**└── UserDao.java           // 用户数据访问**

**├── entity/        // 实体类**

**├── model/     // 数据传输对象**

**├── view/      // 视图对象**

**└── base/      // 基础实体类**

**├── interceptor/   // 拦截器**

**└── AuthorizationInterceptor.java  // 权限拦截器**

**├── service/       // 服务层接口**

**├── impl/      // 服务实现类**

**└── base/      // 基础服务接口**

**└── utils/         // 工具类**

**├── R.java             // 统一返回工具**

**├── PageUtils.java     // 分页工具**

**└── ValidatorUtils.java // 验证工具**

### **3.2 命名规范**

#### 3.2.1 类命名规范

· 类名采用大驼峰命名法（UpperCamelCase）

· Controller类以Controller结尾

· Service类以Service结尾

· 实体类以Entity结尾

· 数据访问类以Dao结尾

· 工具类以Utils结尾

#### 3.2.2 方法命名规范

· 方法名采用小驼峰命名法（lowerCamelCase）

· 查询类方法以get/query/find开头

· 修改类方法以update开头

· 删除类方法以delete/remove开头

· 插入类方法以insert/add/save开头

· 统计类方法以count开头

#### 3.2.3 变量命名规范

· 变量名采用小驼峰命名法

· 常量名全大写，单词间用下划线分隔

· boolean类型变量以is/has/can开头

· 集合命名要体现出复数含义

## **4. 编码规范**

### **4.1 Java编码规范**

#### 4.1.1 注解使用规范

@RestController        *// 控制器注解*

@RequestMapping        *// 请求映射*

@Service              *// 服务层注解*

@Autowired            *// 依赖注入*

@Transactional        *// 事务管理*

@Override             *// 方法重写*

#### 4.1.2 异常处理规范

public class EIException extends RuntimeException {

    private String msg;

    private int code = 500;

    public EIException(String *msg*) {

        super(msg);

        this.msg = msg;

    }

    public EIException(String *msg*, int *code*) {

        super(msg);

        this.msg = msg;

        this.code = code;

    }

}

### **4.2 接口设计规范**

#### 4.2.1 RESTful接口规范

· GET：查询操作

· POST：新增操作

· PUT：更新操作

· DELETE：删除操作

#### 4.2.2 统一返回格式

public class R extends HashMap<String, Object> {

    public static R ok() {

        return new R().put("code", 0);

    }

    public static R error(String *msg*) {

        return error(500, msg);

    }

    public static R ok(String *msg*) {

        return new R().put("msg", msg);

    }

}

### **4.3 数据库规范**

#### 4.3.1 表命名规范

· 使用小写字母，单词间用下划线分隔

· 表名必须是名词

· 表名要能准确反映表的用途

· 临时表以tmp\_开头

#### 4.3.2 字段命名规范

· 使用小写字母，单词间用下划线分隔

· 字段名必须是名词

· 主键统一使用id

· 创建时间统一使用create\_time

· 更新时间统一使用update\_time

## **5. 注释规范**

### **5.1 类注释规范**

***/\*\****

***\* 类说明***

***\****

***\* @author 作者***

***\* @date 创建时间***

***\* @version 版本号***

***\*/***

### **5.2 方法注释规范**

***/\*\****

***\* 方法说明***

***\****

***\* @param 参数名 参数说明***

***\* @return 返回值说明***

***\* @throws 异常类型 异常说明***

***\*/***

### **5.3 代码注释规范**

· 复杂逻辑必须添加注释

· 注释要简洁明了

· 注释要及时更新

· 注释和代码要对齐

## **6. 版本控制规范**

### **6.1 版本号规则**

· 主版本号：重大更新，不兼容旧版本

· 次版本号：功能更新，兼容旧版本

· 修订号：Bug修复，兼容旧版本

### **6.2 分支管理规范**

· master：主分支，稳定版本

· develop：开发分支

· feature：功能分支

· hotfix：紧急修复分支

### **6.3 提交信息规范**

· feat：新功能

· fix：修复bug

· docs：文档更新

· style：代码格式

· refactor：重构

· test：测试相关

## **7. 其他规范**

### **7.1 日志规范**

· 使用统一的日志工具

· 分级记录（INFO/ERROR等）

· 关键操作必须记录日志

· 异常必须记录日志

· 敏感信息脱敏处理

### **7.2 安全规范**

· 密码必须加密存储

· 敏感数据传输必须加密

· SQL注入防护

· XSS防护

· CSRF防护

· 文件上传限制

### **7.3 性能规范**

· 合理使用缓存

· 避免大事务

· 避免循环中查询数据库

· 合理使用索引

· 避免大量数据查询

本规范将根据项目发展持续更新完善，所有开发人员必须严格遵守。