我们的代码结构：

src/main/java/com/

├── annotation/ // 注解包

├── APPLoginUser.java // APP端用户登录注解

├── LoginUser.java // Web端用户登录注解

└── IgnoreAuth.java // 忽略认证注解

├── **config/**

**├── AlipayConfig.java // 支付宝支付配置类**

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

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

├── controller/

├── AddressController.java // 地址管理控制器

├── CartController.java // 购物车控制器

├── ChatController.java // 客服聊天控制器

├── CommonController.java // 通用功能控制器

├── ConfigController.java // 系统配置控制器

├── DiscussershoutushuController.java // 二手图书评论控制器

├── ErshoutushuController.java // 二手图书控制器

├── FileController.java // 文件上传下载控制器

├── MaijiayonghuController.java // 卖家用户控制器

├── MessagesController.java // 留言反馈控制器

├── NewsController.java // 系统公告控制器

├── OrdersController.java // 订单管理控制器

├── QiugouhuifuController.java // 求购回复控制器

├── QiugoutushuController.java // 求购图书控制器

├── StoreupController.java // 收藏管理控制器

├── TushufenleiController.java // 图书分类控制器

├── UserController.java // 用户管理控制器

└── YonghuController.java // 普通用户控制器

│

├── service/

├── AddressService.java // 地址服务接口

├── CartService.java // 购物车服务接口

├── ChatService.java // 客服聊天服务接口

├── CommonService.java // 通用服务接口

├── ConfigService.java // 系统配置服务接口

├── DiscussershoutushuService.java // 二手图书评论服务接口

├── ErshoutushuService.java // 二手图书服务接口

├── MaijiayonghuService.java // 卖家用户服务接口

├── MessagesService.java // 留言反馈服务接口

├── NewsService.java // 系统公告服务接口

├── OrdersService.java // 订单服务接口

├── QiugouhuifuService.java // 求购回复服务接口

├── QiugoutushuService.java // 求购图书服务接口

├── StoreupService.java // 收藏服务接口

├── TokenService.java // Token服务接口

├── TushufenleiService.java // 图书分类服务接口

├── UserService.java // 系统用户服务接口

├── YonghuService.java // 普通用户服务接口

└── src/main/java/com/service/impl/

├── AddressServiceImpl.java // 处理收货地址的增删改查

├── CartServiceImpl.java // 购物车管理，包括添加商品、修改数量、删除商品等

├── ChatServiceImpl.java // 在线客服聊天功能，处理用户咨询和回复

├── CommonServiceImpl.java // 通用功能实现，如选项查询、数据统计

├── ConfigServiceImpl.java // 系统配置管理，处理系统参数设置

├── DiscussershoutushuServiceImpl.java // 二手图书评论管理，包括发表评论、回复

├── ErshoutushuServiceImpl.java // 二手图书管理，包括发布、修改、下架等

├── MaijiayonghuServiceImpl.java // 卖家用户管理，包括注册、登录、信息修改等

├── MessagesServiceImpl.java // 留言反馈管理，处理用户留言和回复

├── NewsServiceImpl.java // 系统公告管理，发布和维护系统公告

├── OrdersServiceImpl.java // 订单管理，包括创建订单、支付、发货、收货等

├── QiugouhuifuServiceImpl.java // 求购回复管理，处理用户求购信息的回复

├── QiugoutushuServiceImpl.java // 求购图书管理，发布和管理求购信息

├── StoreupServiceImpl.java // 收藏管理，处理用户收藏的商品

├── TokenServiceImpl.java // Token管理，处理用户登录认证和会话管

├── TushufenleiServiceImpl.java // 图书分类管理，维护图书分类体系

├── UserServiceImpl.java // 系统用户管理，管理后台用户账号

└── YonghuServiceImpl.java // 普通用户管理，处理普通用户账号相关操

├── dao/

├── AddressDao.java // 地址数据访问接口

├── CartDao.java // 购物车数据访问接口

├── ChatDao.java // 客服聊天数据访问接口

├── CommonDao.java // 通用数据访问接口

├── ConfigDao.java // 系统配置数据访问接口

├── DiscussershoutushuDao.java // 二手图书评论数据访问接口

├── ErshoutushuDao.java // 二手图书数据访问接口

├── MaijiayonghuDao.java // 卖家用户数据访问接口

├── MessagesDao.java // 留言反馈数据访问接口

├── NewsDao.java // 系统公告数据访问接口

├── OrdersDao.java // 订单数据访问接口

├── QiugouhuifuDao.java // 求购回复数据访问接口

├── QiugoutushuDao.java // 求购图书数据访问接口

├── StoreupDao.java // 收藏数据访问接口

├── TokenDao.java // Token数据访问接口

├── TushufenleiDao.java // 图书分类数据访问接口

├── UserDao.java // 系统用户数据访问接口

└── YonghuDao.java // 普通用户数据访问接口

├── entity/

├── AddressEntity.java // 地址实体类

├── CartEntity.java // 购物车实体类

├── ChatEntity.java // 客服聊天实体类

├── ConfigEntity.java // 系统配置实体类

├── DiscussershoutushuEntity.java // 二手图书评论实体类

├── EIException.java // 自定义异常类

├── ErshoutushuEntity.java // 二手图书实体类

├── MaijiayonghuEntity.java // 卖家用户实体类

├── MessagesEntity.java // 留言反馈实体类

├── NewsEntity.java // 系统公告实体类

├── OrdersEntity.java // 订单实体类

├── QiugouhuifuEntity.java // 求购回复实体类

├── QiugoutushuEntity.java // 求购图书实体类

├── StoreupEntity.java // 收藏实体类

├── TokenEntity.java // Token实体类

├── TushufenleiEntity.java // 图书分类实体类

├── UserEntity.java // 系统用户实体类

├── YonghuEntity.java // 普通用户实体类

└── model/ // 数据模型目录

├── AddressModel.java // 地址数据模型

├── CartModel.java // 购物车数据模型

├── ChatModel.java // 聊天数据模型

├── DiscussershoutushuModel.java // 图书评论数据模型

├── ErshoutushuModel.java // 二手图书数据模型

├── MaijiayonghuModel.java // 卖家用户数据模型

├── MessagesModel.java // 留言数据模型

├── NewsModel.java // 公告数据模型

├── OrdersModel.java // 订单数据模型

├── QiugouhuifuModel.java // 求购回复数据模型

├── QiugoutushuModel.java // 求购图书数据模型

├── StoreupModel.java // 收藏数据模型

├── TushufenleiModel.java // 图书分类数据模型

└── YonghuModel.java // 用户数据模型

└── interceptor/

└── AuthorizationInterceptor.java // 权限验证拦截器

└── utils/

├── BaiduUtil.java // 百度API工具类，处理地理位置和认证

├── CommonUtil.java // 通用工具类，提供随机字符串等功能

├── FileUtil.java // 文件处理工具类

├── HttpClientUtils.java // HTTP请求工具类

├── JQPageInfo.java // JQuery分页信息封装类

├── MD5Util.java // MD5加密工具类

├── MPUtil.java // MyBatis-Plus工具类

├── PageUtils.java // 分页工具类

├── Query.java // 查询参数处理类

├── R.java // 统一返回结果封装类

├── SQLFilter.java // SQL注入过滤工具类

├── SpringContextUtils.java // Spring上下文工具类

└── ValidatorUtils.java // 数据验证工具类

└──SpringbootSchemaApplication.java启动类

源代码分析：

1. annotation包：

这个包包含三个注解类，用于处理用户认证相关的功能：

APPLoginUser：APP端用户登录注解

IgnoreAuth：忽略Token验证注解

LoginUser：Web端用户登录注解

### **示例代码分析**

**@Target(ElementType.METHOD)**

**@Retention(RetentionPolicy.RUNTIME)**

**@Documented**

**public @interface IgnoreAuth {**

**}**

### **代码作用分析**

@Target(ElementType.METHOD) 表示此注解只能用于方法

@Retention(RetentionPolicy.RUNTIME) 表示此注解在运行时可用

@Documented 表示此注解应该包含在JavaDoc中

此注解用于标记不需要进行Token验证的接口方法，通常用于登录、注册等公开接口

1. config包：

这个包包含三个注解类，用于处理用户认证相关的功能：

· APPLoginUser：APP端用户登录注解

· IgnoreAuth：忽略Token验证注解

· LoginUser：Web端用户登录注解

### **配置包功能分析**

这个包包含三个配置类，分别处理支付、拦截器和MyBatis-Plus配置。

### **示例代码分析1（AlipayConfig.java）**

***/\* \****

***\*类名：AlipayConfig***

***\*功能：基础配置类***

***\*详细：设置帐户有关信息及返回路径***

***\*/***

**public class AlipayConfig {**

**}**

作用：支付宝支付的配置模板，用于设置商户账户信息和支付回调路径

### **示例代码分析2（InterceptorConfig.java）**

**@Override**

**public void addInterceptors(InterceptorRegistry registry) {**

**registry.addInterceptor(getAuthorizationInterceptor())**

**.addPathPatterns("/\*\*")**

**.excludePathPatterns("/static/\*\*");**

**}**

作用：配置全局权限拦截器，拦截所有非静态资源的请求进行权限验证

### **示例代码分析3（MybatisPlusConfig.java）**

**@Bean**

**public PaginationInterceptor paginationInterceptor() {**

**return new PaginationInterceptor();**

**}**

作用：配置MyBatis-Plus的分页插件，为系统提供分页查询功能支持

1. **controler包：**

包含了系统的所有控制器，负责处理前端请求和返回数据，主要包括：

· **用户相关控制器**

· UserController：系统用户管理

· YonghuController：普通用户管理

· MaijiayonghuController：卖家用户管理

· **图书相关控制器**

· ErshoutushuController：二手图书管理

· TushufenleiController：图书分类管理

· QiugoutushuController：求购图书管理

· QiugouhuifuController：求购回复管理

· **交易相关控制器**

· CartController：购物车管理

· OrdersController：订单管理

· StoreupController：收藏管理

· AddressController：地址管理

· **系统功能控制器**

· CommonController：通用功能

· ConfigController：系统配置

· FileController：文件上传下载

· NewsController：系统公告

### **示例代码分析（OrdersController.java）**

**@RequestMapping("/value/{xColumnName}/{yColumnName}")**

**public R value(@PathVariable("yColumnName") String yColumnName,**

**@PathVariable("xColumnName") String xColumnName,**

**HttpServletRequest request) {**

**Map<String, Object> params = new HashMap<String, Object>();**

**params.put("xColumn", xColumnName);**

**params.put("yColumn", yColumnName);**

**EntityWrapper<OrdersEntity> ew = new EntityWrapper<OrdersEntity>();**

**String tableName = request.getSession().getAttribute("tableName").toString();**

**if(tableName.equals("maijiayonghu")) {**

**ew.eq("maijiazhanghao", (String)request.getSession().getAttribute("username"));**

**}**

**ew.in("status", new String[]{"已支付","已发货","已完成"});**

**List<Map<String, Object>> result = ordersService.selectValue(params, ew);**

**return R.ok().put("data", result);**

**}**

1. Dao包

这段代码展示了订单统计功能的实现，包括：

· 按不同维度统计订单数据

· 根据用户角色过滤数据

· 只统计已完成的订单

· 返回统计结果

包含了所有的数据访问接口，基于MyBatis-Plus实现，主要包括：

· **用户相关DAO**

· UserDao：系统用户数据访问

· YonghuDao：普通用户数据访问

· MaijiayonghuDao：卖家用户数据访问

· **图书相关DAO**

· ErshoutushuDao：二手图书数据访问

· TushufenleiDao：图书分类数据访问

· QiugoutushuDao：求购图书数据访问

· QiugouhuifuDao：求购回复数据访问

· **交易相关DAO**

· CartDao：购物车数据访问

· OrdersDao：订单数据访问

· StoreupDao：收藏数据访问

· AddressDao：地址数据访问

### **示例代码分析（OrdersDao.java）**

**public interface OrdersDao extends BaseMapper<OrdersEntity> {**

**List<OrdersVO> selectListVO(@Param("ew") Wrapper<OrdersEntity> *wrapper*);**

**OrdersVO selectVO(@Param("ew") Wrapper<OrdersEntity> *wrapper*);**

**List<OrdersView> selectListView(@Param("ew") Wrapper<OrdersEntity> *wrapper*);**

**List<OrdersView> selectListView(Pagination *page*,@Param("ew") Wrapper<OrdersEntity> *wrapper*);**

**OrdersView selectView(@Param("ew") Wrapper<OrdersEntity> *wrapper*);**

**List<Map<String, Object>> selectValue(@Param("params") Map<String, Object> *params*,**

**@Param("ew") Wrapper<OrdersEntity> *wrapper*);**

**}**

这段代码展示了典型的DAO接口设计：

· 继承BaseMapper获取基础CRUD功能

· 提供多种查询方法(VO、View等)

· 支持分页查询

· 支持条件构造器

· 支持统计查询

1. entity包：

### **entity包功能分析**

这个包包含了所有的实体类，主要分为三类：基础实体类、Model类和异常类。

· **基础实体类**

· 用户相关：UserEntity、YonghuEntity、MaijiayonghuEntity

· 图书相关：ErshoutushuEntity、TushufenleiEntity

· 交易相关：OrdersEntity、CartEntity、StoreupEntity

· 系统相关：ConfigEntity、NewsEntity

· **Model类**（位于model子包）

· 用于接收前端传参的数据传输对象

· 相比Entity类去掉了一些不必要的字段

· 每个Entity都有对应的Model类

### **示例代码分析（OrdersEntity.java）**

**@TableName("orders")**

**public class OrdersEntity<T> implements Serializable {**

**@TableId**

**private Long id;**

**private String orderid;      *// 订单编号***

**private String tablename;    *// 商品表名***

**private Long userid;         *// 用户id***

**private Long goodid;         *// 商品id***

**private String goodname;     *// 商品名称***

**private Float price;         *// 价格***

**private Float total;         *// 总价格***

**private String status;       *// 状态***

**private String address;      *// 地址***

**private String consignee;    *// 收货人***

**@JsonFormat(locale="zh", timezone="GMT+8", pattern="yyyy-MM-dd HH:mm:ss")**

**@DateTimeFormat**

**private Date addtime;        *// 添加时间***

**}**

这段代码展示了典型的实体类设计：

· 使用MyBatis-Plus注解

· 实现序列化接口

· 包含完整的属性定义

· 提供getter/setter方法

· 支持日期格式化

· 支持泛型转换

1. interceptor包;

包含了系统的拦截器，主要用于处理权限验证和跨域请求。

### **示例代码分析1（跨域请求处理）**

**@Override**

**public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) {**

***//支持跨域请求***

**response.setHeader("Access-Control-Allow-Methods", "POST, GET, OPTIONS, DELETE");**

**response.setHeader("Access-Control-Max-Age", "3600");**

**response.setHeader("Access-Control-Allow-Credentials", "true");**

**response.setHeader("Access-Control-Allow-Headers", "x-requested-with,Token, Origin, Content-Type");**

**response.setHeader("Access-Control-Allow-Origin", request.getHeader("Origin"));**

**if (request.getMethod().equals(RequestMethod.OPTIONS.name())) {**

**response.setStatus(HttpStatus.OK.value());**

**return false;**

**}**

**return true;**

**}**

### **示例代码分析2（Token验证）**

***//从header中获取token***

**String token = request.getHeader(LOGIN\_TOKEN\_KEY);**

**TokenEntity tokenEntity = null;**

**if(StringUtils.isNotBlank(token)) {**

**tokenEntity = tokenService.getTokenEntity(token);**

**}**

**if(tokenEntity != null) {**

**request.getSession().setAttribute("userId", tokenEntity.getUserid());**

**request.getSession().setAttribute("role", tokenEntity.getRole());**

**return true;**

**}**

### **示例代码分析3（权限注解处理）**

**IgnoreAuth annotation;**

**if (handler instanceof HandlerMethod) {**

**annotation = ((HandlerMethod) handler).getMethodAnnotation(IgnoreAuth.class);**

**} else {**

**return true;**

**}**

***//不需要验证权限的方法直接放过***

**if(annotation!=null) {**

**return true;**

**}**

### **示例代码分析4（会话信息设置）**

**if(tokenEntity != null) {**

**request.getSession().setAttribute("userId", tokenEntity.getUserid());**

**request.getSession().setAttribute("role", tokenEntity.getRole());**

**request.getSession().setAttribute("tableName", tokenEntity.getTablename());**

**request.getSession().setAttribute("username", tokenEntity.getUsername());**

**return true;**

**}**

### **示例代码分析5（错误响应处理）**

**PrintWriter writer = null;**

**response.setCharacterEncoding("UTF-8");**

**response.setContentType("application/json; charset=utf-8");**

**try {**

**writer = response.getWriter();**

**writer.print(JSONObject.toJSONString(R.error(401, "请先登录")));**

**} finally {**

**if(writer != null){**

**writer.close();**

**}**

**}**

**return false;**

这些代码展示了拦截器的主要功能：

· 处理跨域请求

· 验证用户Token

· 处理权限注解

· 管理会话信息

· 统一错误处理

· 响应格式化

1. service包：

包含了所有业务逻辑接口及其实现类，基于MyBatis-Plus实现。

### **示例代码分析1（TokenService生成Token）**

**@Override**

**public String generateToken(Long userid, String username, String tableName, String role) {**

**TokenEntity tokenEntity = this.selectOne(**

**new EntityWrapper<TokenEntity>()**

**.eq("userid", userid)**

**.eq("role", role)**

**);**

**String token = CommonUtil.getRandomString(32);**

**Calendar cal = Calendar.getInstance();**

**cal.setTime(new Date());**

**cal.add(Calendar.HOUR\_OF\_DAY, 1);**

**if(tokenEntity != null) {**

**tokenEntity.setToken(token);**

**tokenEntity.setExpiratedtime(cal.getTime());**

**this.updateById(tokenEntity);**

**} else {**

**this.insert(new TokenEntity(userid, username, tableName, role, token, cal.getTime()));**

**}**

**return token;**

**}**

### **示例代码分析2（通用分页查询）**

**@Override**

**public PageUtils queryPage(Map<String, Object> params) {**

**Page<UserEntity> page = this.selectPage(**

**new Query<UserEntity>(params).getPage(),**

**new EntityWrapper<UserEntity>()**

**);**

**return new PageUtils(page);**

**}**

### **示例代码分析3（视图查询）**

**@Override**

**public List<ErshoutushuView> selectListView(Wrapper<ErshoutushuEntity> wrapper) {**

**return baseMapper.selectListView(wrapper);**

**}**

**@Override**

**public ErshoutushuView selectView(Wrapper<ErshoutushuEntity> wrapper) {**

**return baseMapper.selectView(wrapper);**

**}**

### **示例代码分析4（统计查询）**

**@Override**

**public List<Map<String, Object>> selectValue(Map<String, Object> params,**

**Wrapper<OrdersEntity> wrapper) {**

**return baseMapper.selectValue(params, wrapper);**

**}**

**@Override**

**public List<Map<String, Object>> selectTimeStatValue(Map<String, Object> params,**

**Wrapper<OrdersEntity> wrapper) {**

**return baseMapper.selectTimeStatValue(params, wrapper);**

**}**

### **示例代码分析5（通用服务实现）**

**@Service("commonService")**

**public class CommonServiceImpl implements CommonService {**

**@Override**

**public List<String> getOption(Map<String, Object> *params*) {**

**return commonDao.getOption(params);**

**}**

**@Override**

**public Map<String, Object> getFollowByOption(Map<String, Object> *params*) {**

**return commonDao.getFollowByOption(params);**

**}**

**@Override**

**public void sh(Map<String, Object> *params*) {**

**commonDao.sh(params);**

**}**

**}**

这些服务实现了：

· Token管理

· 分页查询

· 视图查询

· 统计分析

· 通用服务

· 实体CRUD操作

1. util包;

包含了系统的各种工具类，用于支持核心业务功能。

### **示例代码分析1（BaiduUtil - 地理位置解析）**

**public static Map<String, String> getCityByLonLat(String key, String lng, String lat) {**

**String location = lat + "," + lng;**

**String url = "http://api.map.baidu.com/reverse\_geocoding/v3/?ak="+key+"&output=json&coordtype=wgs84ll&location="+location;**

**String result = HttpClientUtils.doGet(url);**

**JSONObject o = new JSONObject(result);**

**Map<String, String> area = new HashMap<>();**

**area.put("province", o.getJSONObject("result").getJSONObject("addressComponent").getString("province"));**

**area.put("city", o.getJSONObject("result").getJSONObject("addressComponent").getString("city"));**

**return area;**

**}**

### **示例代码分析2（CommonUtil - 随机字符串生成）**

**public static String getRandomString(Integer num) {**

**String base = "abcdefghijklmnopqrstuvwxyz0123456789";**

**Random random = new Random();**

**StringBuffer sb = new StringBuffer();**

**for (int i = 0; i < num; i++) {**

**int number = random.nextInt(base.length());**

**sb.append(base.charAt(number));**

**}**

**return sb.toString();**

**}**

### **示例代码分析3（SQLFilter - SQL注入防护）**

**public static String sqlInject(String str){**

**if(StringUtils.isBlank(str)){**

**return null;**

**}**

**str = StringUtils.replace(str, "'", "");**

**str = StringUtils.replace(str, "\"", "");**

**str = str.toLowerCase();**

**String[] keywords = {"master", "truncate", "insert", "select"};**

**for(String keyword : keywords){**

**if(str.indexOf(keyword) != -1){**

**throw new EIException("包含非法字符");**

**}**

**}**

**return str;**

**}**

### **示例代码分析4（PageUtils - 分页工具）**

**public PageUtils(List<?> list, int totalCount, int pageSize, int currPage) {**

**this.list = list;**

**this.total = totalCount;**

**this.pageSize = pageSize;**

**this.currPage = currPage;**

**this.totalPage = (int)Math.ceil((double)totalCount/pageSize);**

**}**

### **示例代码分析5（R - 统一返回结果）**

**public static R error(int code, String msg) {**

**R r = new R();**

**r.put("code", code);**

**r.put("msg", msg);**

**return r;**

**}**

**public static R ok(String msg) {**

**R r = new R();**

**r.put("msg", msg);**

**return r;**

**}**

这些工具类提供了：

· 地理位置解析

· 随机字符串生成

· SQL注入防护

· 分页查询支持

· 统一返回结果封装

· 文件处理

· Spring上下文工具

· 数据验证

1. utils 包：

包含了系统的各种工具类，用于支持核心业务功能。

### **示例代码分析1（BaiduUtil - 地理位置解析）**

**public static Map<String, String> getCityByLonLat(String key, String lng, String lat) {**

**String location = lat + "," + lng;**

**String url = "http://api.map.baidu.com/reverse\_geocoding/v3/?ak="+key+"&output=json&coordtype=wgs84ll&location="+location;**

**String result = HttpClientUtils.doGet(url);**

**JSONObject o = new JSONObject(result);**

**Map<String, String> area = new HashMap<>();**

**area.put("province", o.getJSONObject("result").getJSONObject("addressComponent").getString("province"));**

**area.put("city", o.getJSONObject("result").getJSONObject("addressComponent").getString("city"));**

**return area;**

**}**

### **示例代码分析2（CommonUtil - 随机字符串生成）**

**public static String getRandomString(Integer num) {**

**String base = "abcdefghijklmnopqrstuvwxyz0123456789";**

**Random random = new Random();**

**StringBuffer sb = new StringBuffer();**

**for (int i = 0; i < num; i++) {**

**int number = random.nextInt(base.length());**

**sb.append(base.charAt(number));**

**}**

**return sb.toString();**

**}**

### **示例代码分析3（SQLFilter - SQL注入防护）**

**public static String sqlInject(String str){**

**if(StringUtils.isBlank(str)){**

**return null;**

**}**

**str = StringUtils.replace(str, "'", "");**

**str = StringUtils.replace(str, "\"", "");**

**str = str.toLowerCase();**

**String[] keywords = {"master", "truncate", "insert", "select"};**

**for(String keyword : keywords){**

**if(str.indexOf(keyword) != -1){**

**throw new EIException("包含非法字符");**

**}**

**}**

**return str;**

**}**

### **示例代码分析4（PageUtils - 分页工具）**

**public PageUtils(List<?> list, int totalCount, int pageSize, int currPage) {**

**this.list = list;**

**this.total = totalCount;**

**this.pageSize = pageSize;**

**this.currPage = currPage;**

**this.totalPage = (int)Math.ceil((double)totalCount/pageSize);**

**}**

### **示例代码分析5（R - 统一返回结果）**

**public static R error(int code, String msg) {**

**R r = new R();**

**r.put("code", code);**

**r.put("msg", msg);**

**return r;**

**}**

**public static R ok(String msg) {**

**R r = new R();**

**r.put("msg", msg);**

**return r;**

**}**

这些工具类提供了：

· 地理位置解析

· 随机字符串生成

· SQL注入防护

· 分页查询支持

· 统一返回结果封装

· 文件处理

· Spring上下文工具

· 数据验证