# 前端

以下是按照包和所含代码的层级结构进行的文本输出： ### src #### api - \*\*index.js\*\* ```javascript export: (params) => request.get('/activities/export', { params, responseType: 'blob' }) ``` - \*\*message.js\*\* ```javascript import request from './index' // 获取系统消息 export const getSystemMessages = (params) => { return request({ url: '/messages/system', method: 'get', params }) } // 获取活动通知 export const getActivityMessages = (params) => { return request({ url: '/messages/activity', method: 'get', params }) } // 标记消息已读 export const markMessageRead = (messageId) => { return request({ url: `/messages/${messageId}/read`, method: 'put' }) } // 标记所有消息已读 export const markAllMessagesRead = (type) => { return request({ url: `/messages/mark-all-read`, method: 'put', data: { type } }) } // 获取消息设置 export const getMessageSettings = () => { return request({ url: '/messages/settings', method: 'get' }) } // 更新消息设置 export const updateMessageSettings = (data) => { return request({ url: '/messages/settings', method: 'put', data }) } // 获取未读消息数量 export const getUnreadCount = () => { return request({ url: '/messages/unread-count', method: 'get' }) } // 删除消息 export const deleteMessage = (messageId) => { return request({ url: `/messages/${messageId}`, method: 'delete' }) } // 批量删除消息 export const batchDeleteMessages = (messageIds) => { return request({ url: '/messages/batch-delete', method: 'post', data: { messageIds } }) } ``` #### components/common - \*\*FilePreview.vue\*\* ```vue <!-- 部分代码片段展示 --> <el-option label="GB2312" value="gb2312" /> </el-select> </div> <div class="text-container" ref="textContainer"> <pre class="text-content" :style="textStyle">{{ textContent }}</pre> </div> </div> <!-- 代码预览 --> <div v-else-if="isCode" class="code-preview"> <div class="code-toolbar" v-if="showCodeTools"> <el-button-group> <el-button size="small" :icon="ZoomIn" @click="increaseFontSize" /> <el-button size="small" :icon="ZoomOut" @click="decreaseFontSize" /> <el-button size="small" :icon="CopyDocument" @click="copyCode" /> </el-button-group> <el-select v-model="codeTheme" size="small" style="width: 120px;" @change="updateCodeTheme"> <el-option label="默认" value="default" /> <el-option label="暗色" value="dark" /> <el-option label="GitHub" value="github" /> </el-select> </div> <div class="code-container" ref="codeContainer"> <pre class="code-content" :style="codeStyle"><code :class="`language-${codeLanguage}`">{{ codeContent }}</code></pre> </div> </div> <!-- 压缩包预览 --> <div v-else-if="isArchive" class="archive-preview"> <div class="archive-info"> <el-icon class="archive-icon"><FolderOpened /></el-icon> <div class="archive-details"> <!-- 其他代码片段 --> ``` #### config - \*\*index.js\*\* ```javascript // 统一配置管理文件 - 所有链接和配置都可以在这里统一管理 import { footerConfig } from './footer.js' import { navbarConfig } from './navbar.js' // 全局配置对象 export const globalConfig = { // 网站基本信息 site: { name: '中石大志愿云', fullName: '中国石油大学志愿云平台', description: '连接志愿者与公益活动，传递爱心，共建美好社会', version: '1.0.0', author: '志愿云开发团队' }, // API配置 api: { baseURL: process.env.VUE\_APP\_API\_BASE\_URL || 'http://localhost:3000/api', timeout: 10000 }, // 主题配置 theme: { primaryColor: '#c53030', // 中国红主色调 secondaryColor: '#9c1810', accentColor: '#7f1d1d' }, // 导航栏配置 navbar: navbarConfig, // 页脚配置 footer: footerConfig, // 外部链接配置 - 可以统一管理所有外部链接 externalLinks: { // 社交媒体 social: { wechat: 'https://weixin.qq.com', weibo: 'https://weibo.com', qq: 'https://qq.com' }, // 官方网站 official: { university: 'https://www.upc.edu.cn', // 中国石油大学官网 volunteerCenter: 'https://volunteer.upc.edu.cn' // 志愿服务中心 }, // 帮助和支持 support: { help: '/help', contact: '/contact', feedback: '/feedback', terms: '/terms', privacy: '/privacy' } }, // 联系信息配置 contact: { address: '北京市海淀区志愿服务中心', phone: '400-123-4567', email: 'service@zhiyuanyun.com', workingHours: '周一至周五 9:00-18:00' }, // 功能开关配置 features: { enableNotifications: true, // 是否启用通知功能 enableRecommendations: true, // 是否启用智能推荐 enableSocialLogin: true, // 是否启用社交登录 enableDarkMode: false, // 是否启用暗黑模式 enableMultiLanguage: false // 是否启用多语言 } } // 导出各个配置模块 export { footerConfig } from './footer.js' export { navbarConfig } from './navbar.js' // 导出默认配置 export default globalConfig /\*\* \* 配置使用说明： \* \* 1. 统一配置管理： \* import { globalConfig } from '@/config' \* console.log(globalConfig.site.name) \* \* 2. 单独导入配置： \* import { footerConfig, navbarConfig } from '@/config' \* \* 3. 修改配置： \* // 修改网站名称 \* globalConfig.site.name = '新的网站名称' \* \* // 修改主色调 \* globalConfig.theme.primaryColor = '#ff6b35' \* \* // 添加新的外部链接 \* globalConfig.externalLinks.social.douyin = 'https://douyin.com' \* \* 4. 功能开关： \* // 关闭智能推荐功能 \* globalConfig.features.enableRecommendations = false \* \* 5. 环境配置： \* // 在 .env 文件中设置 \* VUE\_APP\_API\_BASE\_URL=https://api.zhiyuanyun.com \* \* 6. 主题定制： \* // 在CSS中使用配置的颜色 \* :root { \* --primary-color: v-bind('globalConfig.theme.primaryColor'); \* } \*/ ``` - \*\*navbar.js\*\*：无完整代码提供 - \*\*footer.js\*\*：无完整代码提供 - \*\*README.md\*\* ```markdown # 配置文件说明 本目录包含了志愿云平台的所有配置文件，所有的链接、导航、页脚信息等都可以通过修改这些配置文件来统一管理。 ## 文件结构 ``` config/ ├── index.js # 统一配置入口文件 ├── navbar.js # 导航栏配置 ├── footer.js # 页脚配置 └── README.md # 本说明文件 ``` ## 配置文件详解 ### 2. navbar.js - 导航栏配置 配置顶部导航栏的所有内容： - \*\*网站标题和Logo\*\* - \*\*主导航菜单\*\*：首页、活动、推荐等 - \*\*用户下拉菜单\*\*：个人中心、设置、退出等 - \*\*默认用户信息\*\* ### 3. footer.js - 页脚配置 配置页面底部的所有内容： - \*\*基本信息\*\*：平台名称、描述 - \*\*社交媒体链接\*\* - \*\*快速导航链接\*\* - \*\*服务支持链接\*\* - \*\*联系信息\*\* - \*\*版权信息\*\* ## 如何修改配置 ### 修改网站名称 ```javascript // 在 index.js 中 globalConfig.site.name = '新的网站名称' // 在 navbar.js 中 navbarConfig.site.title = '新的网站名称' // 在 footer.js 中 footerConfig.title = '新的网站名称' ``` ### 添加新的导航项目 ```javascript // 在 navbar.js 中的 mainNavigation 数组中添加 { index: 'new-page', name: '新页面', icon: 'Document', route: '/new-page', description: '新功能页面' } ``` ### 添加外部链接 ```javascript // 在 footer.js 中的 services 数组中添加 { name: '新的外部链接', url: 'https://example.com', external: true } ``` ### 修改主题颜色 ```javascript // 在 index.js 中 globalConfig.theme.primaryColor = '#新的颜色值' ``` ## 使用配置的组件 以下组件已经配置为使用这些配置文件： - `NavBar.vue` - 使用 `navbar.js` 配置 - `Footer.vue` - 使用 `footer.js` 配置 ## 环境变量 可以通过环境变量来配置某些参数： ```bash # .env 文件 VUE\_APP\_API\_BASE\_URL=https://api.zhiyuanyun.com VUE\_APP\_SITE\_NAME=志愿云平台 ``` ## 示例：完整的配置修改流程 假设要将网站改为"大学生志愿服务平台"： 1. 修改 `index.js`： ```javascript site: { name: '大学生志愿服务平台', fullName: '大学生志愿服务管理平台' } ``` 2. 修改 `navbar.js`： ```javascript site: { title: '大学生志愿服务平台' } ``` 3. 修改 `footer.js`： ```javascript title: '大学生志愿服务平台', description: '连接大学生志愿者与公益活动' ``` 4. 保存文件，页面会自动更新显示新的名称 通过这种方式，可以快速、统一地修改整个网站的配置信息。 ``` #### constants - \*\*index.js\*\* ```javascript /\*\* \* 应用常量定义 \*/ // 用户角色 export const USER\_ROLES = { USER: 'user', ADMIN: 'admin' } // 用户角色标签 export const USER\_ROLE\_LABELS = { [USER\_ROLES.USER]: '普通用户', [USER\_ROLES.ADMIN]: '管理员' } // 用户状态 export const USER\_STATUS = { ACTIVE: 'active', INACTIVE: 'inactive', BANNED: 'banned' } // 用户状态标签 export const USER\_STATUS\_LABELS = { [USER\_STATUS.ACTIVE]: '正常', [USER\_STATUS.INACTIVE]: '未激活', [USER\_STATUS.BANNED]: '已禁用' } // 活动状态 export const ACTIVITY\_STATUS = { DRAFT: 'draft', PUBLISHED: 'published', ONGOING: 'ongoing', COMPLETED: 'completed', CANCELLED: 'cancelled' } // 活动状态标签 export const ACTIVITY\_STATUS\_LABELS = { [ACTIVITY\_STATUS.DRAFT]: '草稿', [ACTIVITY\_STATUS.PUBLISHED]: '已发布', [ACTIVITY\_STATUS.ONGOING]: '进行中', [ACTIVITY\_STATUS.COMPLETED]: '已完成', [ACTIVITY\_STATUS.CANCELLED]: '已取消' } // 活动状态颜色 export const ACTIVITY\_STATUS\_COLORS = { [ACTIVITY\_STATUS.DRAFT]: '#909399', [ACTIVITY\_STATUS.PUBLISHED]: '#409EFF', [ACTIVITY\_STATUS.ONGOING]: '#67C23A', [ACTIVITY\_STATUS.COMPLETED]: '#E6A23C', [ACTIVITY\_STATUS.CANCELLED]: '#F56C6C' } // 活动类别 export const ACTIVITY\_CATEGORIES = { EDUCATION: 'education', ENVIRONMENT: 'environment', COMMUNITY: 'community', ELDERLY: 'elderly', CHILDREN: 'children', DISABLED: 'disabled', POVERTY: 'poverty', DISASTER: 'disaster', CULTURE: 'culture', SPORTS: 'sports', HEALTH: 'health', OTHER: 'other' } // 活动类别标签 export const ACTIVITY\_CATEGORY\_LABELS = { [ACTIVITY\_CATEGORIES.EDUCATION]: '教育支教', [ACTIVITY\_CATEGORIES.ENVIRONMENT]: '环境保护', [ACTIVITY\_CATEGORIES.COMMUNITY]: '社区服务', [ACTIVITY\_CATEGORIES.ELDERLY]: '关爱老人', [ACTIVITY\_CATEGORIES.CHILDREN]: '关爱儿童', [ACTIVITY\_CATEGORIES.DISABLED]: '助残服务', [ACTIVITY\_CATEGORIES.POVERTY]: '扶贫济困', [ACTIVITY\_CATEGORIES.DISASTER]: '应急救援', [ACTIVITY\_CATEGORIES.CULTURE]: '文化宣传', [ACTIVITY\_CATEGORIES.SPORTS]: '体育健身', [ACTIVITY\_CATEGORIES.HEALTH]: '医疗健康', [ACTIVITY\_CATEGORIES.OTHER]: '其他' } // 活动类别颜色 export const ACTIVITY\_CATEGORY\_COLORS = { [ACTIVITY\_CATEGORIES.EDUCATION]: '#409EFF', [ACTIVITY\_CATEGORIES.ENVIRONMENT]: '#67C23A', [ACTIVITY\_CATEGORIES.COMMUNITY]: '#E6A23C', [ACTIVITY\_CATEGORIES.ELDERLY]: '#F56C6C', [ACTIVITY\_CATEGORIES.CHILDREN]: '#FF69B4', [ACTIVITY\_CATEGORIES.DISABLED]: '#9370DB', [ACTIVITY\_CATEGORIES.POVERTY]: '#32CD32', [ACTIVITY\_CATEGORIES.DISASTER]: '#FF4500', [ACTIVITY\_CATEGORIES.CULTURE]: '#20B2AA', [ACTIVITY\_CATEGORIES.SPORTS]: '#FFD700', [ACTIVITY\_CATEGORIES.HEALTH]: '#FF6347', [ACTIVITY\_CATEGORIES.OTHER]: '#909399' } // 报名状态 export const SIGNUP\_STATUS = { PENDING: 'pending', APPROVED: 'approved', REJECTED: 'rejected', CANCELLED: 'cancelled', CHECKED\_IN: 'checked\_in', COMPLETED: 'completed' } // 报名状态标签 export const SIGNUP\_STATUS\_LABELS = { [SIGNUP\_STATUS.PENDING]: '待审核', [SIGNUP\_STATUS.APPROVED]: '已通过', [SIGNUP\_STATUS.REJECTED]: '已拒绝', [SIGNUP\_STATUS.CANCELLED]: '已取消', [SIGNUP\_STATUS.CHECKED\_IN]: '已签到', [SIGNUP\_STATUS.COMPLETED]: '已完成' } // 报名状态颜色 export const SIGNUP\_STATUS\_COLORS = { [SIGNUP\_STATUS.PENDING]: '#E6A23C', [SIGNUP\_STATUS.APPROVED]: '#67C23A', [SIGNUP\_STATUS.REJECTED]: '#F56C6C', [SIGNUP\_STATUS.CANCELLED]: '#909399', [SIGNUP\_STATUS.CHECKED\_IN]: '#409EFF', [SIGNUP\_STATUS.COMPLETED]: '#67C23A' } // 公告类型 export const ANNOUNCEMENT\_TYPES = { SYSTEM: 'system', ACTIVITY: 'activity', NOTICE: 'notice', NEWS: 'news' } // 公告类型标签 export const ANNOUNCEMENT\_TYPE\_LABELS = { [ANNOUNCEMENT\_TYPES.SYSTEM]: '系统公告', [ANNOUNCEMENT\_TYPES.ACTIVITY]: '活动公告', [ANNOUNCEMENT\_TYPES.NOTICE]: '通知公告', [ANNOUNCEMENT\_TYPES.NEWS]: '新闻资讯' } // 公告状态 export const ANNOUNCEMENT\_STATUS = { DRAFT: 'draft', PUBLISHED: 'published', OFFLINE: 'offline' } // 公告状态标签 export const ANNOUNCEMENT\_STATUS\_LABELS = { [ANNOUNCEMENT\_STATUS.DRAFT]: '草稿', [ANNOUNCEMENT\_STATUS.PUBLISHED]: '已发布', [ANNOUNCEMENT\_STATUS.OFFLINE]: '已下线' } // 性别 export const GENDER = { MALE: 'male', FEMALE: 'female', OTHER: 'other' } // 性别标签 export const GENDER\_LABELS = { [GENDER.MALE]: '男', [GENDER.FEMALE]: '女', [GENDER.OTHER]: '其他' } // 文件类型 export const FILE\_TYPES = { IMAGE: 'image', DOCUMENT: 'document', VIDEO: 'video', AUDIO: 'audio', OTHER: 'other' } // 允许的图片类型 export const ALLOWED\_IMAGE\_TYPES = [ 'image/jpeg', 'image/jpg', 'image/png', 'image/gif', 'image/webp' ] // 允许的文档类型 export const ALLOWED\_DOCUMENT\_TYPES = [ 'application/pdf', 'application/msword', 'application/vnd.openxmlformats-officedocument.wordprocessingml.document', 'application/vnd.ms-excel', 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet', 'application/vnd.ms-powerpoint', 'application/vnd.openxmlformats-officedocument.presentationml.presentation', 'text/plain' ] // 文件大小限制（字节） export const FILE\_SIZE\_LIMITS = { IMAGE: 10 \* 1024 \* 1024, // 10MB DOCUMENT: 50 \* 1024 \* 1024, // 50MB VIDEO: 100 \* 1024 \* 1024, // 100MB AUDIO: 20 \* 1024 \* 1024 // 20MB } // 分页配置 export const PAGINATION = { DEFAULT\_PAGE\_SIZE: 10, PAGE\_SIZES: [10, 20, 50, 100], MAX\_PAGE\_SIZE: 100 } // 排序方向 export const SORT\_ORDER = { ASC: 'asc', DESC: 'desc' } // 排序方向标签 export const SORT\_ORDER\_LABELS = { [SORT\_ORDER.ASC]: '升序', [SORT\_ORDER.DESC]: '降序' } // 时间格式 export const DATE\_FORMATS = { DATE: 'YYYY-MM-DD', DATETIME: 'YYYY-MM-DD HH:mm:ss', TIME: 'HH:mm:ss', MONTH: 'YYYY-MM', YEAR: 'YYYY' } // 推荐频率 export const RECOMMEND\_FREQUENCY = { DAILY: 'daily', WEEKLY: 'weekly', MONTHLY: 'monthly' } // 推荐频率标签 export const RECOMMEND\_FREQUENCY\_LABELS = { [RECOMMEND\_FREQUENCY.DAILY]: '每日推荐', [RECOMMEND\_FREQUENCY.WEEKLY]: '每周推荐', [RECOMMEND\_FREQUENCY.MONTHLY]: '每月推荐' } // 时间偏好 export const TIME\_PREFERENCES = { WEEKDAY: 'weekday', WEEKEND: 'weekend', BOTH: 'both' } // 时间偏好标签 export const TIME\_PREFERENCE\_LABELS = { [TIME\_PREFERENCES.WEEKDAY]: '工作日', [TIME\_PREFERENCES.WEEKEND]: '周末', [TIME\_PREFERENCES.BOTH]: '不限' } // AI服务类型 export const AI\_SERVICE\_TYPES = { RECOMMEND: 'recommend', GENERATE: 'generate', ANALYZE: 'analyze' } // AI服务类型标签 export const AI\_SERVICE\_TYPE\_LABELS = { [AI\_SERVICE\_TYPES.RECOMMEND]: '智能推荐', [AI\_SERVICE\_TYPES.GENERATE]: '内容生成', [AI\_SERVICE\_TYPES.ANALYZE]: '数据分析' } // AI操作类型 export const AI\_OPERATION\_TYPES = { GET\_RECOMMENDATIONS: 'get\_recommendations', UPDATE\_PREFERENCES: 'update\_preferences', FEEDBACK: 'feedback', GENERATE\_CONTENT: 'generate\_content', ANALYZE\_DATA: 'analyze\_data' } // AI操作类型标签 export const AI\_OPERATION\_TYPE\_LABELS = { [AI\_OPERATION\_TYPES.GET\_RECOMMENDATIONS]: '获取推荐', [AI\_OPERATION\_TYPES.UPDATE\_PREFERENCES]: '更新偏好', [AI\_OPERATION\_TYPES.FEEDBACK]: '反馈', [AI\_OPERATION\_TYPES.GENERATE\_CONTENT]: '生成内容', [AI\_OPERATION\_TYPES.ANALYZE\_DATA]: '分析数据' } // 响应状态码 export const HTTP\_STATUS = { OK: 200, CREATED: 201, NO\_CONTENT: 204, BAD\_REQUEST: 400, UNAUTHORIZED: 401, FORBIDDEN: 403, NOT\_FOUND: 404, INTERNAL\_SERVER\_ERROR: 500 } // 本地存储键名 export const STORAGE\_KEYS = { TOKEN: 'token', USER: 'user', THEME: 'theme', LANGUAGE: 'language', SIDEBAR\_COLLAPSED: 'sidebar\_collapsed', SEARCH\_HISTORY: 'search\_history', PREFERENCES: 'preferences' } // 主题 export const THEMES = { LIGHT: 'light', DARK: 'dark', AUTO: 'auto' } // 主题标签 export const THEME\_LABELS = { [THEMES.LIGHT]: '浅色主题', [THEMES.DARK]: '深色主题', [THEMES.AUTO]: '跟随系统' } // 语言 export const LANGUAGES = { ZH\_CN: 'zh-cn', EN\_US: 'en-us' } // 语言标签 export const LANGUAGE\_LABELS = { [LANGUAGES.ZH\_CN]: '简体中文', [LANGUAGES.EN\_US]: 'English' } // 默认配置 export const DEFAULT\_CONFIG = { PAGE\_SIZE: PAGINATION.DEFAULT\_PAGE\_SIZE, THEME: THEMES.LIGHT, LANGUAGE: LANGUAGES.ZH\_CN, DEBOUNCE\_DELAY: 300, REQUEST\_TIMEOUT: 10000 } // 正则表达式 export const REGEX\_PATTERNS = { EMAIL: /^[^\s@]+@[^\s@]+\.[^\s@]+$/, PHONE: /^1[3-9]\d{9}$/, ID\_CARD: /(^\d{15}$)|(^\d{18}$)|(^\d{17}(\d|X|x)$)/, USERNAME: /^[a-zA-Z][a-zA-Z0-9\_]{3,19}$/, PASSWORD: /^(?=.\*[a-zA-Z])(?=.\*\d)[a-zA-Z\d!@#$%^&\*()\_+\-=\[\]{};':"\\|,.<>\/?]{8,}$/, URL: /^https?:\/\/(www\.)?[-a-zA-Z0-9@:%.\_\+~#=]{1,256}\.[a-zA-Z0-9()]{1,6}\b([-a-zA-Z0-9()@:%\_\+.~#?&//=]\*)$/, IP: /^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/, CHINESE: /[\u4e00-\u9fa5]/, NUMBER: /^\d+$/, DECIMAL: /^\d+(\.\d+)?$/ } export default { USER\_ROLES, USER\_ROLE\_LABELS, USER\_STATUS, USER\_STATUS\_LABELS, ACTIVITY\_STATUS, ACTIVITY\_STATUS\_LABELS, ACTIVITY\_STATUS\_COLORS, ACTIVITY\_CATEGORIES, ACTIVITY\_CATEGORY\_LABELS, ACTIVITY\_CATEGORY\_COLORS, SIGNUP\_STATUS, SIGNUP\_STATUS\_LABELS, SIGNUP\_STATUS\_COLORS, ANNOUNCEMENT\_TYPES, ANNOUNCEMENT\_TYPE\_LABELS, ANNOUNCEMENT\_STATUS, ANNOUNCEMENT\_STATUS\_LABELS, GENDER, GENDER\_LABELS, FILE\_TYPES, ALLOWED\_IMAGE\_TYPES, ALLOWED\_DOCUMENT\_TYPES, FILE\_SIZE\_LIMITS, } ``` #### router - \*\*index.js\*\* ```javascript const Recommend = () => import('@/views/user/Recommend.vue') ``` #### styles - \*\*index.css\*\* ```css .fade-enter-active, .fade-leave-active { transition: opacity 0.3s; } .fade-enter-from, .fade-leave-to { opacity: 0; } /\* 统一字体样式 \*/ /\* 标题样式 \*/ h1, h2, h3, h4, h5, h6 { font-weight: 700; letter-spacing: 0.8px; text-shadow: 0 1px 2px rgba(0, 0, 0, 0.1); color: #333; } h1 { font-size: 2.5em; font-weight: 800; letter-spacing: 1.5px; text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.15); } h2 { font-size: 2em; font-weight: 800; letter-spacing: 1.2px; } h3 { font-size: 1.5em; font-weight: 700; letter-spacing: 1px; } h4 { font-size: 1.25em; font-weight: 700; letter-spacing: 0.8px; } /\* 按钮统一样式 \*/ .el-button { font-weight: 600; letter-spacing: 0.5px; text-shadow: 0 1px 1px rgba(0, 0, 0, 0.1); } .el-button--primary { background: #c53030; border-color: #c53030; color: white; font-weight: 700; letter-spacing: 0.8px; text-shadow: 1px 2px 3px rgba(0, 0, 0, 0.2); } .el-button--primary:hover { background: #a02626; border-color: #a02626; } /\* 文本样式 \*/ p, span, div { font-weight: 500; letter-spacing: 0.3px; } /\* 强调文本 \*/ .text-bold { font-weight: 700; letter-spacing: 0.8px; text-shadow: 0 1px 2px rgba(0, 0, 0, 0.1); } /\* 数字统计样式 \*/ .stat-number { font-weight: 800; letter-spacing: 1px; text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2); } /\* 标签样式 \*/ .el-tag { /\* 其他代码片段 \*/ } ``` #### utils - \*\*index.js\*\* ```javascript /\*\* \* 通用工具函数 \*/ /\*\* \* 防抖函数 \* @param {Function} func 要防抖的函数 \* @param {number} delay 延迟时间 \* @returns {Function} 防抖后的函数 \*/ export function debounce(func, delay = 300) { let timeoutId return function (...args) { clearTimeout(timeoutId) timeoutId = setTimeout(() => func.apply(this, args), delay) } } /\*\* \* 节流函数 \* @param {Function} func 要节流的函数 \* @param {number} delay 延迟时间 \* @returns {Function} 节流后的函数 \*/ export function throttle(func, delay = 300) { let lastTime = 0 return function (...args) { const now = Date.now() if (now - lastTime >= delay) { lastTime = now func.apply(this, args) } } } /\*\* \* 深拷贝 \* @param {any} obj 要拷贝的对象 \* @returns {any} 拷贝后的对象 \*/ export function deepClone(obj) { if (obj === null || typeof obj !== 'object') { return obj } if (obj instanceof Date) { return new Date(obj.getTime()) } if (obj instanceof Array) { return obj.map(item => deepClone(item)) } if (typeof obj === 'object') { const clonedObj = {} for (const key in obj) { if (obj.hasOwnProperty(key)) { clonedObj[key] = deepClone(obj[key]) } } return clonedObj } } /\*\* \* 生成唯一ID \* @param {number} length ID长度 \* @returns {string} 唯一ID \*/ export function generateId(length = 8) { const chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789' let result = '' for (let i = 0; i < length; i++) { result += chars.charAt(Math.floor(Math.random() \* chars.length)) } return result } /\*\* \* 格式化文件大小 \* @param {number} bytes 字节数 \* @returns {string} 格式化后的文件大小 \*/ export function formatFileSize(bytes) { if (bytes === 0) return '0 B' const k = 1024 const sizes = ['B', 'KB', 'MB', 'GB', 'TB'] const i = Math.floor(Math.log(bytes) / Math.log(k)) return parseFloat((bytes / Math.pow(k, i)).toFixed(2)) + ' ' + sizes[i] } /\*\* \* 下载文件 \* @param {Blob|string} data 文件数据或URL \* @param {string} filename 文件名 \*/ export function downloadFile(data, filename) { const link = document.createElement('a') if (typeof data === 'string') { link.href = data } else { const url = window.URL.createObjectURL(data) link.href = url } link.download = filename document.body.appendChild(link) link.click() document.body.removeChild(link) if (typeof data !== 'string') { window.URL.revokeObjectURL(link.href) } } /\*\* \* 复制文本到剪贴板 \* @param {string} text 要复制的文本 \* @returns {Promise<boolean>} 是否复制成功 \*/ export async function copyToClipboard(text) { try { if (navigator.clipboard && window.isSecureContext) { await navigator.clipboard.writeText(text) return true } else { // 降级方案 const textArea = document.createElement('textarea') textArea.value = text textArea.style.position = 'fixed' textArea.style.left = '-999999px' textArea.style.top = '-999999px' document.body.appendChild(textArea) textArea.focus() textArea.select() const result = document.execCommand('copy') document.body.removeChild(textArea) return result } } catch (error) { console.error('Copy to clipboard failed:', error) return false } } /\*\* \* 获取URL参数 \* @param {string} name 参数名 \* @param {string} url URL字符串 \* @returns {string|null} 参数值 \*/ export function getUrlParam(name, url = window.location.href) { const urlObj = new URL(url) return urlObj.searchParams.get(name) } /\*\* \* 设置URL参数 \* @param {string} name 参数名 \* @param {string} value 参数值 \* @param {string} url URL字符串 \* @returns {string} 新的URL \*/ export function setUrlParam(name, value, url = window.location.href) { const urlObj = new URL(url) urlObj.searchParams.set(name, value) return urlObj.toString() } /\*\* \* 删除URL参数 \* @param {string} name 参数名 \* @param {string} url URL字符串 \* @returns {string} 新的URL \*/ export function removeUrlParam(name, url = window.location.href) { const urlObj = new URL(url) urlObj.searchParams.delete(name) return urlObj.toString() } /\*\* \* 判断是否为移动设备 \* @returns {boolean} 是否为移动设备 \*/ export function isMobile() { return /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera Mini/i.test(navigator.userAgent) } /\*\* \* 获取设备类型 \* @returns {string} 设备类型 \*/ export function getDeviceType() { const ua = navigator.userAgent if (/tablet|ipad|playbook|silk/i.test(ua)) { return 'tablet' } if (/mobile|iphone|ipod|android|blackberry|opera|mini|windows\sce|palm|smartphone|iemobile/i.test(ua)) { return 'mobile' } return 'desktop' } /\*\* \* 滚动到指定元素 \* @param {string|Element} element 元素选择器或元素 \* @param {object} options 滚动选项 \*/ export function scrollToElement(element, options = {}) { const targetElement = typeof element === 'string' ? document.querySelector(element) : element if (targetElement) { targetElement.scrollIntoView({ behavior: 'smooth', block: 'start', inline: 'nearest', ...options }) } } /\*\* \* 滚动到顶部 \* @param {number} duration 动画持续时间 \*/ export function scrollToTop(duration = 300) { const start = window.pageYOffset const startTime = performance.now() function scroll() { const now = performance.now() const time = Math.min(1, (now - startTime) / duration) const timeFunction = 1 - Math.pow(1 - time, 3) // easeOutCubic window.scrollTo(0, Math.ceil(start \* (1 - timeFunction))) if (time < 1) { requestAnimationFrame(scroll) } } requestAnimationFrame(scroll) } /\*\* \* 获取元素相对于视口的位置 \* @param {Element} element 元素 \* @returns {object} 位置信息 \*/ export function getElementPosition(element) { const rect = element.getBoundingClientRect() return { top: rect.top, left: rect.left, right: rect.right, bottom: rect.bottom, width: rect.width, height: rect.height, centerX: rect.left + rect.width / 2, centerY: rect.top + rect.height / 2 } } /\*\* \* 判断元素是否在视口中 \* @param {Element} element 元素 \* @param {number} threshold 阈值（0-1） \* @returns {boolean} 是否在视口中 \*/ export function isElementInViewport(element, threshold = 0) { const rect = element.getBoundingClientRect() const windowHeight = window.innerHeight || document.documentElement.clientHeight const windowWidth = window.innerWidth || document.documentElement.clientWidth const vertInView = (rect.top <= windowHeight \* (1 - threshold)) && ((rect.top + rect.height) >= windowHeight \* threshold) const horInView = (rect.left <= windowWidth \* (1 - threshold)) && ((rect.left + rect.width) >= windowWidth \* threshold) return vertInView && horInView } /\*\* \* 等待指定时间 \* @param {number} ms 等待时间（毫秒） \* @returns {Promise} Promise对象 \*/ export function sleep(ms) { return new Promise(resolve => setTimeout(resolve, ms)) } /\*\* \* 重试函数 \* @param {Function} fn 要重试的函数 \* @param {number} maxRetries 最大重试次数 \* @param {number} delay 重试间隔 \* @returns {Promise} Promise对象 \*/ export async function retry(fn, maxRetries = 3, delay = 1000) { let lastError for (let i = 0; i <= maxRetries; i++) { try { return await fn() } catch (error) { lastError = error if (i < maxRetries) { await sleep(delay) } } } throw lastError } /\*\* \* 对象转换为FormData \* @param {object} obj 对象 \* @returns {FormData} FormData对象 \*/ export function objectToFormData(obj) { const formData = new FormData() function appendFormData(data, root) { if (data instanceof File) { formData.append(root, data) } else if (Array.isArray(data)) { data.forEach((item, index) => { appendFormData(item, `${root}[${index}]`) }) } else if (typeof data === 'object' && data !== null) { Object.keys(data).forEach(key => { appendFormData(data[key], root ? `${root}[${key}]` : key) }) } else { formData.append(root, data) } } appendFormData(obj) return formData } /\*\* \* 数组去重 \* @param {Array} arr 数组 \* @param {string|Function} key 去重键或函数 \* @returns {Array} 去重后的数组 \*/ export function uniqueArray(arr, key) { if (!key) { return [...new Set(arr)] } const seen = new Set() return arr.filter(item => { const value = typeof key === 'function' ? key(item) : item[key] if (seen.has(value)) { return false } seen.add(value) return true }) } /\*\* \* 数组分组 \* @param {Array} arr 数组 \* @param {string|Function} key 分组键或函数 \* @returns {object} 分组后的对象 \*/ export function groupBy(arr, key) { return arr.reduce((groups, item) => { const value = typeof key === 'function' ? key(item) : item[key] if (!groups[value]) { groups[value] = [] } groups[value].push(item) return groups }, {}) } /\*\* \* 数组排序 \* @param {Array} arr 数组 \* @param {string|Function} key 排序键或函数 \* @param {string} order 排序方向 'asc' | 'desc' \* @returns {Array} 排序后的数组 \*/ export function sortBy(arr, key, order = 'asc') { return [...arr].sort((a, b) => { const valueA = typeof key === 'function' ? key(a) : a[key] // 其他代码片段 }) } ``` #### views/admin - \*\*Logs.vue\*\* ```vue <!-- 部分代码片段展示 --> link.download = `${activeTab.value}\_logs\_${new Date().toISOString().split('T')[0]}.xlsx` link.click() window.URL.revokeObjectURL(url) ElMessage.success('导出成功') } catch (error) { ElMessage.error('导出失败') } finally { exporting.value = false } } const getActionType = (action) => { const typeMap = { login: 'success', logout: 'info', create: 'success', update: 'warning', delete: 'danger', review: 'primary' } return typeMap[action] || 'info' } const getActionText = (action) => { const textMap = { login: '登录', logout: '登出', create: '创建', update: '更新', delete: '删除', review: '审核' } return textMap[action] || action } const getLevelType = (level) => { const typeMap = { info: 'info', warning: 'warning', error: 'danger', debug: 'success' } return typeMap[level] || 'info' } const getLevelText = (level) => { const textMap = { info: '信息', warning: '警告', error: '错误', debug: '调试' } return textMap[level] || level } const getFieldLabel = (key) => { const labelMap = { username: '用户名', action: '操作', resource: '资源', description: '描述', ip: 'IP地址', userAgent: '用户代理', status: '状态', location: '地点', device: '设备', failReason: '失败原因', <!-- 其他代码片段 --> ``` #### main.js ```javascript import { createApp } from 'vue' import { createPinia } from 'pinia' import ElementPlus from 'element-plus' import 'element-plus/dist/index.css' import \* as ElementPlusIconsVue from '@element-plus/icons-vue' import App from './App.vue' import router from './router' import './styles/index.css' import './styles/red-theme.css' const app = createApp(App) const pinia = createPinia() // 注册所有图标 for (const [key, component] of Object.entries(ElementPlusIconsVue)) { app.component(key, component) } app.use(pinia) app.use(router) app.use(ElementPlus) app.mount('#app') ```

# 后端

以下是根据代码库中的代码整理的项目层级结构： ### com.cupk - \*\*BackendApplication.java\*\* ```java package com.cupk; import com.cupk.service.PasswordMigrationService; import org.mybatis.spring.annotation.MapperScan; import org.springframework.boot.CommandLineRunner; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean; import org.springframework.transaction.annotation.EnableTransactionManagement; @SpringBootApplication @MapperScan("com.cupk.mapper") @EnableTransactionManagement public class BackendApplication { public static void main(String[] args) { SpringApplication.run(BackendApplication.class, args); } @Bean public CommandLineRunner run(PasswordMigrationService migrationService) { return args -> { System.out.println("===== 开始密码迁移到MD5 ====="); migrationService.migrateToMD5(); System.out.println("===== 密码迁移完成 ====="); }; } } ``` ### com.cupk.config - \*\*JwtFilterConfig.java\*\* ```java import com.cupk.filter.JwtAuthenticationFilter; import org.springframework.boot.web.servlet.FilterRegistrationBean; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.core.Ordered; @Configuration public class JwtFilterConfig { @Bean public FilterRegistrationBean<JwtAuthenticationFilter> jwtFilter( JwtAuthenticationFilter filter) { FilterRegistrationBean<JwtAuthenticationFilter> registration = new FilterRegistrationBean<>(); registration.setFilter(filter); registration.addUrlPatterns("/api/\*"); // 只保护API路径 registration.setOrder(Ordered.HIGHEST\_PRECEDENCE + 1); return registration; } } ``` - \*\*PasswordEncoderConfig.java\*\* ```java package com.cupk.config; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.security.crypto.password.MessageDigestPasswordEncoder; import org.springframework.security.crypto.password.PasswordEncoder; @Configuration public class PasswordEncoderConfig { @Bean public PasswordEncoder passwordEncoder() { // 使用 MD5 加密 return new MessageDigestPasswordEncoder("MD5"); } } ``` - \*\*MybatisPlusConfig.java\*\* ```java package com.cupk.config; import com.baomidou.mybatisplus.annotation.DbType; import com.baomidou.mybatisplus.autoconfigure.ConfigurationCustomizer; import com.baomidou.mybatisplus.extension.plugins.MybatisPlusInterceptor; import com.baomidou.mybatisplus.extension.plugins.inner.PaginationInnerInterceptor; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; @Configuration public class MybatisPlusConfig { @Bean public MybatisPlusInterceptor mybatisPlusInterceptor() { MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor(); // 分页插件 interceptor.addInnerInterceptor(new PaginationInnerInterceptor(DbType.MYSQL)); return interceptor; } // // 添加缺失的配置 // @Bean // public ConfigurationCustomizer configurationCustomizer() { // return configuration -> configuration.setUseDeprecatedExecutor(false); // } } ``` - \*\*CorsConfig.java\*\* ```java package com.cupk.config; import org.springframework.context.annotation.Configuration; import org.springframework.web.servlet.config.annotation.CorsRegistry; import org.springframework.web.servlet.config.annotation.WebMvcConfigurer; @Configuration public class CorsConfig implements WebMvcConfigurer { @Override public void addCorsMappings(CorsRegistry registry) { registry.addMapping("/\*\*") .allowedOrigins("\*") .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS") .allowedHeaders("\*") .maxAge(3600); } } ``` - \*\*SecurityConfig.java\*\* ```java package com.cupk.config; import com.cupk.filter.JwtAuthenticationFilter; import com.cupk.service.IUserService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.http.HttpMethod; import org.springframework.security.authentication.AuthenticationManager; import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder; import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration; import org.springframework.security.config.annotation.web.builders.HttpSecurity; import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity; import org.springframework.security.config.http.SessionCreationPolicy; import org.springframework.security.core.userdetails.UserDetailsService; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.security.web.SecurityFilterChain; import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter; @Configuration @EnableWebSecurity public class SecurityConfig { private final IUserService userService; private final PasswordEncoder passwordEncoder; // 使用MD5编码器 private final JwtAuthenticationFilter jwtAuthenticationFilter; @Autowired public SecurityConfig( IUserService userService, PasswordEncoder passwordEncoder, // 直接注入MD5编码器 JwtAuthenticationFilter jwtAuthenticationFilter ) { this.userService = userService; this.passwordEncoder = passwordEncoder; this.jwtAuthenticationFilter = jwtAuthenticationFilter; } @Bean public AuthenticationManager authenticationManager( AuthenticationConfiguration authConfig ) throws Exception { return authConfig.getAuthenticationManager(); } @Autowired public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception { auth.userDetailsService(userService) .passwordEncoder(passwordEncoder); } @Bean public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception { http .csrf(csrf -> csrf.disable()) .authorizeHttpRequests(auth -> auth .requestMatchers("/api/auth/\*\*").permitAll() .requestMatchers("/api/activities").permitAll() .requestMatchers("/api/utils/\*\*").permitAll() // 添加这一行 .requestMatchers("/api/users/\*\*").hasRole("管理员") .requestMatchers(HttpMethod.GET, "/api/users/{id}").authenticated() .requestMatchers("/api/organizations/\*\*").hasAnyRole("组织者", "管理员") .requestMatchers("/api/registrations").authenticated() .requestMatchers(HttpMethod.POST, "/api/registrations").authenticated() .requestMatchers(HttpMethod.GET, "/api/registrations/user/{userId}").authenticated() .anyRequest().authenticated() ) .sessionManagement(session -> session .sessionCreationPolicy(SessionCreationPolicy.STATELESS) ) .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class); return http.build(); } } ``` ### com.cupk.controller - \*\*ServiceRecordController.java\*\* ```java package com.cupk.controller; import com.cupk.entity.ServiceRecord; import com.cupk.service.IServiceRecordService; import com.cupk.utils.ResponseResult; import jakarta.validation.Valid; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.web.bind.annotation.\*; import java.util.List; @RestController @RequestMapping("/api/records") public class ServiceRecordController { private final IServiceRecordService serviceRecordService; @Autowired public ServiceRecordController(IServiceRecordService serviceRecordService) { this.serviceRecordService = serviceRecordService; } @GetMapping @PreAuthorize("hasRole('ADMIN')") public ResponseResult<List<ServiceRecord>> getServiceRecords() { return ResponseResult.success(serviceRecordService.list()); } @GetMapping("/user/{userId}") @PreAuthorize("hasRole('ADMIN') or #userId == authentication.name") public ResponseResult<List<ServiceRecord>> getServiceRecordsByUserId(@PathVariable String userId) { List<ServiceRecord> records = serviceRecordService.lambdaQuery() .eq(ServiceRecord::getUserid, userId) .list(); return ResponseResult.success(records); } @PostMapping @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Boolean> createServiceRecord(@Valid @RequestBody ServiceRecord serviceRecord) { boolean saved = serviceRecordService.save(serviceRecord); if (saved) { return ResponseResult.success(true); } else { // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(500, "创建失败", false); } } @PutMapping("/{id}") @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Boolean> updateServiceRecord(@PathVariable Long id, @Valid @RequestBody ServiceRecord serviceRecord) { serviceRecord.setId(id); boolean updated = serviceRecordService.updateById(serviceRecord); if (updated) { return ResponseResult.success(true); } else { // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(500, "更新失败", false); } } @DeleteMapping("/{id}") @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Boolean> deleteServiceRecord(@PathVariable Long id) { boolean removed = serviceRecordService.removeById(id); if (removed) { return ResponseResult.success(true); } else { // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(500, "删除失败", false); } } } ``` - \*\*ActivityController.java\*\* ```java package com.cupk.controller; import com.cupk.entity.Activity; import com.cupk.service.IActivityService; import com.cupk.utils.ResponseResult; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.web.bind.annotation.\*; @RestController @RequestMapping("/api/activities") public class ActivityController extends BaseController<Activity> { private final IActivityService activityService; @Autowired public ActivityController(IActivityService activityService) { this.activityService = activityService; } @GetMapping("/{id}") public ResponseResult<Activity> getActivityById(@PathVariable Long id) { Activity activity = activityService.getById(id); if (activity != null) { return getDataResult(activity); } else { // 使用明确指定类型的错误响应 return ResponseResult.error(404, "活动不存在"); } } @PutMapping("/{id}") @PreAuthorize("hasRole('ORGANIZER') or hasRole('ADMIN')") public ResponseResult<Activity> updateActivity(@PathVariable Long id, @RequestBody Activity activity) { activity.setId(id); boolean updated = activityService.updateById(activity); if (updated) { return getDataResult(activity); } else { // 返回正确的类型 - 使用 null 表示无数据 return new ResponseResult<>(500, "更新失败", null); } } @DeleteMapping("/{id}") @PreAuthorize("hasRole('ORGANIZER') or hasRole('ADMIN')") public ResponseResult<String> deleteActivity(@PathVariable Long id) { boolean removed = activityService.removeById(id); if (removed) { return successResult("活动删除成功"); } else { return ResponseResult.error("删除失败"); } } @PostMapping @PreAuthorize("hasRole('ORGANIZER') or hasRole('ADMIN')") public ResponseResult<Activity> createActivity(@RequestBody Activity activity) { activity.setStatus("未开始"); // 设置默认状态 boolean saved = activityService.save(activity); if (saved) { return getDataResult(activity); } return new ResponseResult<>(500, "创建失败", null); } } ``` - \*\*OrganizationController.java\*\* ```java package com.cupk.controller; import com.baomidou.mybatisplus.core.metadata.IPage; import com.baomidou.mybatisplus.extension.plugins.pagination.Page; import com.cupk.entity.Organization; import com.cupk.service.IOrganizationService; import com.cupk.utils.ResponseResult; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.web.bind.annotation.\*; @RestController @RequestMapping("/api/organizations") public class OrganizationController { private final IOrganizationService organizationService; @Autowired public OrganizationController(IOrganizationService organizationService) { this.organizationService = organizationService; } @GetMapping public ResponseResult<IPage<Organization>> getOrganizations( @RequestParam(defaultValue = "1") int page, @RequestParam(defaultValue = "10") int size) { Page<Organization> pageParam = new Page<>(page, size); IPage<Organization> orgPage = organizationService.page(pageParam); return ResponseResult.success(orgPage); } @PostMapping @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Boolean> createOrganization(@RequestBody Organization organization) { boolean saved = organizationService.save(organization); if (saved) { return ResponseResult.success(true); } else { // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(500, "创建失败", false); } } @PutMapping("/{id}") @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Boolean> updateOrganization( @PathVariable String id, @RequestBody Organization organization) { organization.setId(id); boolean updated = organizationService.updateById(organization); if (updated) { return ResponseResult.success(true); } else { // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(500, "更新失败", false); } } @DeleteMapping("/{id}") @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Boolean> deleteOrganization(@PathVariable String id) { boolean removed = organizationService.removeById(id); if (removed) { return ResponseResult.success(true); } else { // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(500, "删除失败", false); } } } ``` - \*\*AnnouncementController.java\*\* ```java package com.cupk.controller; import com.baomidou.mybatisplus.core.metadata.IPage; import com.baomidou.mybatisplus.extension.plugins.pagination.Page; import com.cupk.entity.Announcement; import com.cupk.service.IAnnouncementService; import com.cupk.utils.ResponseResult; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.web.bind.annotation.\*; @RestController @RequestMapping("/api/announcements") public class AnnouncementController { private final IAnnouncementService announcementService; @Autowired public AnnouncementController(IAnnouncementService announcementService) { this.announcementService = announcementService; } @GetMapping public ResponseResult<IPage<Announcement>> getAnnouncements( @RequestParam(defaultValue = "1") int page, @RequestParam(defaultValue = "10") int size, @RequestParam(required = false) String status) { Page<Announcement> pageParam = new Page<>(page, size); IPage<Announcement> announcements = announcementService.getAnnouncements(pageParam, status); return ResponseResult.success(announcements); } @GetMapping("/{id}") public ResponseResult<Announcement> getAnnouncementById(@PathVariable Long id) { Announcement announcement = announcementService.getById(id); if (announcement != null) { return ResponseResult.success(announcement); } else { // 返回正确的类型 - 使用 null 表示无数据 return new ResponseResult<>(404, "公告不存在", null); } } @PostMapping @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Announcement> createAnnouncement(@RequestBody Announcement announcement) { announcement.setStatus("已发布"); if (announcementService.save(announcement)) { return ResponseResult.success(announcement); } else { // 返回正确的类型 - 使用 null 表示无数据 return new ResponseResult<>(500, "发布失败", null); } } @PutMapping("/{id}") @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Announcement> updateAnnouncement( @PathVariable Long id, @RequestBody Announcement announcement) { announcement.setId(id); if (announcementService.updateById(announcement)) { return ResponseResult.success(announcement); } else { // 返回正确的类型 - 使用 null 表示无数据 return new ResponseResult<>(500, "更新失败", null); } } @PutMapping("/{id}/withdraw") @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Boolean> withdrawAnnouncement(@PathVariable Long id) { Announcement announcement = announcementService.getById(id); if (announcement != null) { announcement.setStatus("撤回"); boolean updated = announcementService.updateById(announcement); if (updated) { return ResponseResult.success(true); } else { // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(500, "撤回失败", false); } } // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(404, "公告不存在", false); } @DeleteMapping("/{id}") @PreAuthorize("hasRole('ADMIN')") public ResponseResult<Boolean> deleteAnnouncement(@PathVariable Long id) { boolean removed = announcementService.removeById(id); if (removed) { return ResponseResult.success(true); } else { // 返回正确的类型 - 使用 false 表示操作失败 return new ResponseResult<>(500, "删除失败", false); } } } ``` - \*\*UserController.java\*\* ```java package com.cupk.controller; import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; import com.cupk.entity.User; import com.cupk.service.IUserService; import com.cupk.utils.ResponseResult; import jakarta.validation.Valid; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*; import java.time.LocalDateTime; @RestController @RequestMapping("/api/users") public class UserController extends BaseController<User> { private final IUserService userService; @Autowired public UserController(IUserService userService) { this.userService = userService; } @GetMapping("/{id}") public ResponseResult<User> getUserById(@PathVariable String id) { User user = userService.getById(id); if (user != null) { // 清除敏感信息 user.setPassword(null); return ResponseResult.success(user); } else { return ResponseResult.error(404, "用户不存在"); } } @PostMapping("/register") public ResponseResult<User> registerUser(@RequestBody @Valid User user) { // 检查用户ID是否已存在 if (userService.findByUsername(user.getId()) != null) { return ResponseResult.error(400, "用户ID已存在"); } // 添加邮箱唯一性检查 QueryWrapper<User> emailQuery = new QueryWrapper<>(); emailQuery.eq("email", user.getEmail()); if (userService.getOne(emailQuery) != null) { return ResponseResult.error(400, "邮箱已被注册"); } user.setRegistertime(LocalDateTime.now()); user.setStatus("正常"); userService.save(user); // 返回前清除密码 user.setPassword(null); return getDataResult(user); } } ``` - \*\*AuthController.java\*\* ```java package com.cupk.controller; import com.cupk.entity.User; import com.cupk.utils.JwtUtils; import com.cupk.utils.ResponseResult; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.authentication.AuthenticationManager; import org.springframework.security.authentication.UsernamePasswordAuthenticationToken; import org.springframework.security.core.Authentication; import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestBody; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController; import java.util.Map; @RestController @RequestMapping("/api/auth") public class AuthController { private final AuthenticationManager authenticationManager; private final JwtUtils jwtUtils; @Autowired public AuthController(AuthenticationManager authenticationManager, JwtUtils jwtUtils) { this.authenticationManager = authenticationManager; this.jwtUtils = jwtUtils; } @PostMapping("/login") public ResponseResult<String> login(@RequestBody Map<String, String> loginData) { try { String userId = loginData.get("id"); String password = loginData.get("password"); System.out.println("尝试登录用户: " + userId); System.out.println("提交的密码: " + password); if (userId == null || userId.isBlank()) { return ResponseResult.error(400, "用户ID不能为空"); } Authentication authentication = authenticationManager.authenticate( new UsernamePasswordAuthenticationToken(userId, password) ); System.out.println("认证成功！权限: " + authentication.getAuthorities()); String token = jwtUtils.generateToken(userId); return ResponseResult.success("登录成功", token); } catch (Exception e) { System.err.println("===== 认证失败详情 ====="); e.printStackTrace(); return ResponseResult.error(500, "登录失败: " + e.getMessage()); } } } ``` - \*\*RegistrationController.java\*\* ```java package com.cupk.controller; import com.baomidou.mybatisplus.core.metadata.IPage; import com.baomidou.mybatisplus.extension.plugins.pagination.Page; import com.cupk.entity.Registration; import com.cupk.service.IRegistrationService; import com.cupk.utils.ResponseResult; import jakarta.validation.Valid; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.web.bind.annotation.\*; import java.time.LocalDateTime; import java.util.List; @RestController @RequestMapping("/api/registrations") public class RegistrationController { private final IRegistrationService registrationService; @Autowired public RegistrationController(IRegistrationService registrationService) { this.registrationService = registrationService; } // 用户报名活动 @PostMapping @PreAuthorize("isAuthenticated()") public ResponseResult<Boolean> registerActivity(@Valid @RequestBody Registration registration) { // 设置报名时间为当前时间 registration.setRegistertime(LocalDateTime.now()); // 状态默认为"待审核" registration.setStatus("待审核"); boolean saved = registrationService.save(registration); if (saved) { return ResponseResult.success(true); } else { return ResponseResult.error(500, "报名失败"); } } // 获取用户报名记录 @GetMapping("/user/{userId}") @PreAuthorize("hasRole('ADMIN') or #userId == authentication.name") public ResponseResult<List<Registration>> getRegistrationsByUserId(@PathVariable String userId) { List<Registration> registrations = registrationService.findByUserId(userId); return ResponseResult.success(registrations); } // 获取活动报名记录 @GetMapping("/activity/{activityId}") @PreAuthorize("hasRole('ORGANIZER') or hasRole('ADMIN')") public ResponseResult<List<Registration>> getRegistrationsByActivityId(@PathVariable Long activityId) { List<Registration> registrations = registrationService.findByActivityId(activityId); return ResponseResult.success(registrations); } // 审核报名 @PutMapping("/{id}/review") @PreAuthorize("hasRole('ORGANIZER') or hasRole('ADMIN')") public ResponseResult<Boolean> reviewRegistration(@PathVariable Long id, @RequestParam String status, @RequestParam(required = false) String remark) { Registration registration = registrationService.getById(id); if (registration == null) { return ResponseResult.error(404, "报名记录不存在"); } // 更新状态和备注 registration.setStatus(status); if (remark != null) { registration.setRemark(remark); } boolean updated = registrationService.updateById(registration); if (updated) { return ResponseResult.success(true); } else { return ResponseResult.error(500, "审核失败"); } } // 分页查询报名记录 @GetMapping @PreAuthorize("hasRole('ADMIN')") public ResponseResult<IPage<Registration>> getRegistrations( @RequestParam(defaultValue = "1") int page, @RequestParam(defaultValue = "10") int size) { Page<Registration> pageParam = new Page<>(page, size); IPage<Registration> registrations = registrationService.page(pageParam); return ResponseResult.success(registrations); } } ``` - \*\*BaseController.java\*\* ```java package com.cupk.controller; import com.baomidou.mybatisplus.core.metadata.IPage; import com.cupk.utils.ResponseResult; public class BaseController<T> { protected ResponseResult<IPage<T>> getPageResult(IPage<T> page) { return ResponseResult.success(page); } protected ResponseResult<T> getDataResult(T data) { return ResponseResult.success(data); } protected ResponseResult<String> successResult(String message) { return new ResponseResult<>(200, message, null); } } ``` - \*\*PasswordCheckController.java\*\* ```java package com.cupk.controller; import com.cupk.utils.ResponseResult; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController; @RestController @RequestMapping("/api/utils") public class PasswordCheckController { @Autowired private PasswordEncoder passwordEncoder; @GetMapping("/check-password/{userId}") public ResponseResult<String> checkPassword(@PathVariable String userId) { // 这只是示例，实际需要从数据库获取密码 String md5Password = "e10adc3949ba59abbe56e057f20f883e"; // "123456"的MD5 return ResponseResult.success( "密码检查结果", "用户: " + userId + "\nMD5密码: " + md5Password ); } } ``` ### com.cupk.entity - \*\*Tag.java\*\* ```java package com.cupk.entity; import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName; import lombok.Data; @Data @TableName("tags") public class Tag { @TableId(type = IdType.AUTO) private Long id; private String name; } ``` - \*\*User.java\*\* ```java package com.cupk.entity; import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName; import com.fasterxml.jackson.annotation.JsonFormat; import com.fasterxml.jackson.annotation.JsonIgnore; import com.fasterxml.jackson.annotation.JsonInclude; import lombok.Data; import java.time.LocalDateTime; @Data @TableName("users") // 添加字段忽略策略 @JsonInclude(JsonInclude.Include.NON\_NULL) public class User { @TableId(type = IdType.INPUT) private String id; private String username; private String password; private String realname; private String email; private String phone; private String gender; private String role; private String department; // 添加的字段 private String grade; // 添加的字段 private String avatar = "default.jpg"; @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") private LocalDateTime registertime; private String status = "正常"; } ``` - \*\*UserTag.java\*\* ```java package com.cupk.entity; import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName; import lombok.Data; @Data @TableName("usertags") public class UserTag { @TableId(type = IdType.AUTO) private Long id; private String userid; private Long tagid; } ``` - \*\*Activity.java\*\* ```java package com.cupk.entity; import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName; import com.fasterxml.jackson.annotation.JsonFormat; import lombok.Data; import java.time.LocalDateTime; @Data @TableName("activities") public class Activity { @TableId(type = IdType.AUTO) private Long id; private String title; private String description; private String location; @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") private LocalDateTime starttime; @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") private LocalDateTime endtime; private Integer maxparticipants; private String organizerid; // 负责人ID private String organizationid; // 主办单位ID private String type; private String status = "未开始"; // enum('未开始','进行中','已结束') private String coverimage = "default.jpg"; private String volunteerrequirements; } ``` - \*\*Organization.java\*\* ```java package com.cupk.entity; import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName; import com.fasterxml.jackson.annotation.JsonFormat; import lombok.Data; import java.time.LocalDateTime; @Data @TableName("organizations") public class Organization { @TableId(type = IdType.INPUT) private String id; private String name; private String description; private String contactperson; private String contactphone; private String email; private String address; @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") private LocalDateTime createtime; } ``` - \*\*ServiceRecord.java\*\* ```java package com.cupk.entity; import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName; import com.fasterxml.jackson.annotation.JsonFormat; import lombok.Data; import java.time.LocalDateTime; @Data @TableName("servicerecords") public class ServiceRecord { @TableId(type = IdType.AUTO) private Long id; private String userid; private Long activityid; private Float servicehours; private String content; @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") private LocalDateTime recordtime; private String certifierid; private Float serviceevaluation; } ``` - \*\*Registration.java\*\* ```java package com.cupk.entity; import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName; import com.fasterxml.jackson.annotation.JsonFormat; import lombok.Data; import java.time.LocalDateTime; @Data @TableName("registrations") public class Registration { @TableId(type = IdType.AUTO) private Long id; private String userid; private Long activityid; @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") private LocalDateTime registertime; private String status = "待审核"; // enum('待审核','通过','拒绝') private String remark; } ``` - \*\*Announcement.java\*\* ```java package com.cupk.entity; import com.baomidou.mybatisplus.annotation.IdType; import com.baomidou.mybatisplus.annotation.TableId; import com.baomidou.mybatisplus.annotation.TableName; import com.fasterxml.jackson.annotation.JsonFormat; import lombok.Data; import java.time.LocalDateTime; @Data @TableName("announcements") public class Announcement { @TableId(type = IdType.AUTO) private Long id; private String title; private String content; @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss") private LocalDateTime publishtime; private String publisherid; private String status = "草稿"; // enum('已发布','草稿','撤回') } ``` ### com.cupk.exception - \*\*BusinessException.java\*\* ```java package com.cupk.exception; public class BusinessException extends RuntimeException { public BusinessException(String message) { super(message); } } ``` - \*\*GlobalExceptionHandler.java\*\* ```java // src/main/java/com/cupk/exception/GlobalExceptionHandler.java package com.cupk.exception; import com.cupk.utils.ResponseResult; import org.springframework.http.HttpStatus; import org.springframework.security.access.AccessDeniedException; import org.springframework.web.bind.annotation.ExceptionHandler; import org.springframework.web.bind.annotation.ResponseStatus; import org.springframework.web.bind.annotation.RestControllerAdvice; @RestControllerAdvice public class GlobalExceptionHandler { // 处理权限不足异常 @ExceptionHandler(AccessDeniedException.class) @ResponseStatus(HttpStatus.FORBIDDEN) public ResponseResult<String> handleAccessDeniedException(AccessDeniedException e) { return ResponseResult.error(403, "权限不足"); } // 处理业务异常 @ExceptionHandler(BusinessException.class) @ResponseStatus(HttpStatus.BAD\_REQUEST) public ResponseResult<String> handleBusinessException(BusinessException e) { return ResponseResult.error(400, e.getMessage()); } // 处理其他所有异常 @ExceptionHandler(Exception.class) @ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR) public ResponseResult<String> handleException(Exception e) { return ResponseResult.error(500, "服务器内部错误: " + e.getMessage()); } } ``` ### com.cupk.filter - \*\*JwtAuthenticationFilter.java\*\* ```java package com.cupk.filter; import com.cupk.service.IUserService; import com.cupk.utils.JwtUtils; import jakarta.servlet.FilterChain; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import org.hibernate.validator.internal.util.stereotypes.Lazy; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.authentication.UsernamePasswordAuthenticationToken; import org.springframework.security.core.context.SecurityContextHolder; import org.springframework.security.core.userdetails.UserDetails; import org.springframework.security.web.authentication.WebAuthenticationDetailsSource; import org.springframework.stereotype.Component; import org.springframework.web.filter.OncePerRequestFilter; import jakarta.servlet.FilterChain; import jakarta.servlet.ServletException; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import java.io.IOException; @Component public class JwtAuthenticationFilter extends OncePerRequestFilter { @Lazy @Autowired private JwtUtils jwtUtils; @Autowired private IUserService userService; @Override protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain) throws ServletException, IOException { final String authorizationHeader = request.getHeader("Authorization"); String username = null; String jwt = null; if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) { jwt = authorizationHeader.substring(7); username = jwtUtils.extractUsername(jwt); } if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) { UserDetails userDetails = this.userService.loadUserByUsername(username); if (jwtUtils.validateToken(jwt, userDetails)) { UsernamePasswordAuthenticationToken authentication = new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities()); authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request)); SecurityContextHolder.getContext().setAuthentication(authentication); } } chain.doFilter(request, response); } } ``` ### com.cupk.mapper - \*\*RegistrationMapper.java\*\* ```java package com.cupk.mapper; import org.apache.ibatis.annotations.Mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.cupk.entity.Registration; @Mapper public interface RegistrationMapper extends BaseMapper<Registration> {} ``` - \*\*AnnouncementMapper.java\*\* ```java package com.cupk.mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.cupk.entity.Announcement; import org.apache.ibatis.annotations.Mapper; @Mapper public interface AnnouncementMapper extends BaseMapper<Announcement> {} ``` - \*\*OrganizationMapper.java\*\* ```java package com.cupk.mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.cupk.entity.Organization; import org.apache.ibatis.annotations.Mapper; @Mapper public interface OrganizationMapper extends BaseMapper<Organization> {} ``` - \*\*UserMapper.java\*\* ```java package com.cupk.mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.cupk.entity.User; import org.apache.ibatis.annotations.Mapper; @Mapper public interface UserMapper extends BaseMapper<User> {} ``` - \*\*ActivityMapper.java\*\* ```java package com.cupk.mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.cupk.entity.Activity; import org.apache.ibatis.annotations.Mapper; @Mapper public interface ActivityMapper extends BaseMapper<Activity> {} ``` - \*\*TagMapper.java\*\* ```java package com.cupk.mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.cupk.entity.Tag; import org.apache.ibatis.annotations.Mapper; @Mapper public interface TagMapper extends BaseMapper<Tag> {} ``` - \*\*UserTagMapper.java\*\* ```java package com.cupk.mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.cupk.entity.UserTag; import org.apache.ibatis.annotations.Mapper; @Mapper public interface UserTagMapper extends BaseMapper<UserTag> {} ``` - \*\*ServiceRecordMapper.java\*\* ```java package com.cupk.mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.cupk.entity.ServiceRecord; import org.apache.ibatis.annotations.Mapper; @Mapper public interface ServiceRecordMapper extends BaseMapper<ServiceRecord> {} ``` ### com.cupk.service - \*\*IActivityService.java\*\* ```java package com.cupk.service; import com.baomidou.mybatisplus.extension.service.IService; import com.cupk.entity.Activity; public interface IActivityService extends IService<Activity> { } ``` - \*\*IUserService.java\*\* ```java package com.cupk.service; import com.baomidou.mybatisplus.extension.service.IService; import com.cupk.entity.User; import org.springframework.security.core.userdetails.UserDetailsService; public interface IUserService extends IService<User>, UserDetailsService { User findByUsername(String username); } ``` - \*\*IRegistrationService.java\*\* ```java package com.cupk.service; import com.baomidou.mybatisplus.extension.service.IService; import com.cupk.entity.Registration; import java.util.List; public interface IRegistrationService extends IService<Registration> { List<Registration> findByUserId(String userId); List<Registration> findByActivityId(Long activityId); } ``` - \*\*IServiceRecordService.java\*\* ```java package com.cupk.service; import com.baomidou.mybatisplus.extension.service.IService; import com.cupk.entity.ServiceRecord; public interface IServiceRecordService extends IService<ServiceRecord> {} ``` - \*\*IOrganizationService.java\*\* ```java package com.cupk.service; import com.baomidou.mybatisplus.extension.service.IService; import com.cupk.entity.Organization; public interface IOrganizationService extends IService<Organization> { } ``` - \*\*IAnnouncementService.java\*\* ```java package com.cupk.service; import com.baomidou.mybatisplus.core.metadata.IPage; import com.baomidou.mybatisplus.extension.plugins.pagination.Page; import com.baomidou.mybatisplus.extension.service.IService; import com.cupk.entity.Announcement; public interface IAnnouncementService extends IService<Announcement> { IPage<Announcement> getAnnouncements(Page<Announcement> pageParam, String status); } ``` - \*\*PasswordMigrationService.java\*\* ```java package com.cupk.service; import com.cupk.entity.User; import com.cupk.mapper.UserMapper; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service; import org.springframework.transaction.annotation.Transactional; import java.security.MessageDigest; import java.util.List; @Service public class PasswordMigrationService { private final UserMapper userMapper; @Autowired public PasswordMigrationService(UserMapper userMapper) { this.userMapper = userMapper; } @Transactional public void migrateToMD5() { // 获取所有用户 List<User> users = userMapper.selectList(null); System.out.println("找到用户数量: " + users.size()); for (User user : users) { // 打印当前密码信息 System.out.println("迁移用户: " + user.getId() + ", 原密码: " + user.getPassword() + ", 长度: " + user.getPassword().length()); // 生成 MD5 密码 String md5Password = md5("123456"); // 假设所有用户密码都是123456 System.out.println("新密码(MD5): " + md5Password); // 更新数据库 user.setPassword(md5Password); userMapper.updateById(user); System.out.println("用户密码已更新"); } } // MD5 加密方法 private String md5(String input) { try { java.security.MessageDigest md = java.security.MessageDigest.getInstance("MD5"); byte[] array = md.digest(input.getBytes()); StringBuilder sb = new StringBuilder(); for (byte b : array) { sb.append(String.format("%02x", b)); } return sb.toString(); } catch (java.security.NoSuchAlgorithmException e) { throw new RuntimeException("MD5加密失败", e); } } } ``` ### com.cupk.service.impl - \*\*ActivityServiceImpl.java\*\* ```java package com.cupk.service.impl; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.cupk.entity.Activity; import com.cupk.mapper.ActivityMapper; import com.cupk.service.IActivityService; import org.springframework.stereotype.Service; @Service public class ActivityServiceImpl extends ServiceImpl<ActivityMapper, Activity> implements IActivityService {} ``` - \*\*AnnouncementServiceImpl.java\*\* ```java package com.cupk.service.impl; import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; import com.baomidou.mybatisplus.core.metadata.IPage; import com.baomidou.mybatisplus.extension.plugins.pagination.Page; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.cupk.entity.Announcement; import com.cupk.mapper.AnnouncementMapper; import com.cupk.service.IAnnouncementService; import org.springframework.stereotype.Service; @Service public class AnnouncementServiceImpl extends ServiceImpl<AnnouncementMapper, Announcement> implements IAnnouncementService { @Override public IPage<Announcement> getAnnouncements(Page<Announcement> pageParam, String status) { QueryWrapper<Announcement> wrapper = new QueryWrapper<>(); if (status != null) { wrapper.eq("status", status); } return baseMapper.selectPage(pageParam, wrapper); } } ``` - \*\*OrganizationServiceImpl.java\*\* ```java package com.cupk.service.impl; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.cupk.entity.Organization; import com.cupk.mapper.OrganizationMapper; import com.cupk.service.IOrganizationService; import org.springframework.stereotype.Service; @Service public class OrganizationServiceImpl extends ServiceImpl<OrganizationMapper, Organization> implements IOrganizationService {} ``` - \*\*RegistrationServiceImpl.java\*\* ```java package com.cupk.service.impl; import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.cupk.entity.Registration; import com.cupk.mapper.RegistrationMapper; import com.cupk.service.IRegistrationService; import org.springframework.stereotype.Service; import java.util.List; @Service public class RegistrationServiceImpl extends ServiceImpl<RegistrationMapper, Registration> implements IRegistrationService { @Override public List<Registration> findByUserId(String userId) { QueryWrapper<Registration> wrapper = new QueryWrapper<>(); wrapper.eq("userid", userId); return baseMapper.selectList(wrapper); } @Override public List<Registration> findByActivityId(Long activityId) { QueryWrapper<Registration> wrapper = new QueryWrapper<>(); wrapper.eq("activityid", activityId); return baseMapper.selectList(wrapper); } } ``` - \*\*UserServiceImpl.java\*\* ```java package com.cupk.service.impl; import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.cupk.entity.User; import com.cupk.mapper.UserMapper; import com.cupk.service.IUserService; import org.springframework.security.core.GrantedAuthority; import org.springframework.security.core.authority.SimpleGrantedAuthority; import org.springframework.security.core.userdetails.UserDetails; import org.springframework.security.core.userdetails.UsernameNotFoundException; import org.springframework.stereotype.Service; import java.util.ArrayList; import java.util.List; @Service public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements IUserService { @Override public User findByUsername(String username) { QueryWrapper<User> wrapper = new QueryWrapper<>(); wrapper.eq("id", username); return getOne(wrapper); } @Override public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException { User user = findByUsername(username); if (user == null) { throw new UsernameNotFoundException("用户不存在: " + username); } // 添加日志 System.out.println("加载用户: " + username); System.out.println("数据库密码: " + user.getPassword()); System.out.println("密码长度: " + user.getPassword().length()); // 创建带 ROLE\_ 前缀的权限 List<GrantedAuthority> authorities = new ArrayList<>(); authorities.add(new SimpleGrantedAuthority("ROLE\_" + user.getRole())); return new org.springframework.security.core.userdetails.User( user.getId(), user.getPassword(), // 存储的是 MD5 值 authorities ); } } ``` - \*\*ServiceRecordServiceImpl.java\*\* ```java package com.cupk.service.impl; import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl; import com.cupk.entity.ServiceRecord; import com.cupk.mapper.ServiceRecordMapper; import com.cupk.service.IServiceRecordService; import org.springframework.stereotype.Service; @Service public class ServiceRecordServiceImpl extends ServiceImpl<ServiceRecordMapper, ServiceRecord> implements IServiceRecordService { } ``` ### com.cupk.utils - \*\*JwtUtils.java\*\* ```java package com.cupk.utils; import io.jsonwebtoken.Claims; import io.jsonwebtoken.Jwts; import io.jsonwebtoken.SignatureAlgorithm; import org.springframework.beans.factory.annotation.Value; import org.springframework.security.core.userdetails.UserDetails; import org.springframework.stereotype.Component; import java.util.Date; import java.util.HashMap; import java.util.Map; import java.util.function.Function; import io.jsonwebtoken.security.Keys; import javax.crypto.SecretKey; @Component public class JwtUtils { @Value("${jwt.expiration}") private Long expiration; @Value("${jwt.secret}") private String secretBase64; // 改为Base64编码的密钥 private SecretKey getSigningKey() { byte[] keyBytes = java.util.Base64.getDecoder().decode(secretBase64); return Keys.hmacShaKeyFor(keyBytes); } public String generateToken(String username) { Map<String, Object> claims = new HashMap<>(); return createToken(claims, username); } private String createToken(Map<String, Object> claims, String subject) { return Jwts.builder() .setClaims(claims) .setSubject(subject) .setIssuedAt(new Date(System.currentTimeMillis())) .setExpiration(new Date(System.currentTimeMillis() + expiration \* 1000)) .signWith(getSigningKey()) // 使用安全的签名密钥 .compact(); } public String extractUsername(String token) { return extractClaim(token, Claims::getSubject); } public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) { final Claims claims = extractAllClaims(token); return claimsResolver.apply(claims); } private Claims extractAllClaims(String token) { return Jwts.parserBuilder() .setSigningKey(getSigningKey()) .build() .parseClaimsJws(token) .getBody(); } public Boolean validateToken(String token, UserDetails userDetails) { final String username = extractUsername(token); return (username.equals(userDetails.getUsername()) && !isTokenExpired(token)); } private Boolean isTokenExpired(String token) { return extractExpiration(token).before(new Date()); } public Date extractExpiration(String token) { return extractClaim(token, Claims::getExpiration); } } ``` - \*\*ResponseResult.java\*\* ```java package com.cupk.utils; import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor; @Data @NoArgsConstructor @AllArgsConstructor public class ResponseResult<T> { private int code; private String message; private T data; public static <T> ResponseResult<T> success(T data) { return new ResponseResult<>(200, "成功", data); } public static <T> ResponseResult<T> success(String message, T data) { return new ResponseResult<>(200, message, data); } public static ResponseResult<String> error(String message) { return new ResponseResult<>(500, message, null); } public static <T> ResponseResult<T> error(int code, String message) { return new ResponseResult<>(code, message, null); } } ``` ### src/test/java/com.cupk - \*\*BackendApplicationTests.java\*\* ```java package com.cupk; import org.junit.jupiter.api.Test; import org.springframework.boot.test.context.SpringBootTest; @SpringBootTest class BackendApplicationTests { @Test void contextLoads() { } } ``` ### src/main/resources - \*\*application.yml\*\* ```yaml # JWT配置 jwt: secret: KzI6ZC8lcp0s8+u1SMmcQ5qnCIe3adp6Yr5GUKtO8CQ= expiration: 86400 # 24小时 # 数据库配置 spring: datasource: url: jdbc:mysql://localhost:3306/cupkvolunteercloud?useSSL=false&serverTimezone=UTC username: root password: 123456 driver-class-name: com.mysql.cj.jdbc.Driver main: allow-circular-references: true security: filter: dispatcher-types: REQUEST # 只处理REQUEST类型的过滤器 # MyBatis Plus配置 mybatis-plus: mapper-locations: classpath\*:mapper/\*\*/\*.xml type-aliases-package: com.cupk.entity configuration: map-underscore-to-camel-case: true # 开启驼峰映射 log-impl: org.apache.ibatis.logging.stdout.StdOutImpl # 开启SQL日志 ```