# 1. 登录

## 1.1 Shiro配置

因为是前后端分离项目，Shiro默认支持的是*前后端不分离*的场景，并且认证机鉴权是基于Cookie的，因此我们需要对Shiro中的部分类进行重写。jsp

### 重写sessionId接收方式

知识备注：

　　从Spring3.0，@Configuration用于定义配置类，可替换xml配置文件，被注解的类内部包含有一个或多个被@Bean注解的方法，这些方法将会被AnnotationConfigApplicationContext或AnnotationConfigWebApplicationContext类进行扫描，并用于构建bean定义，初始化Spring容器。

注意：@Configuration注解的配置类有如下要求：

@Configuration不可以是final类型；

@Configuration不可以是匿名类；

嵌套的configuration必须是静态类。

正常来讲 Shiro 是从 Cookie 中获取 SessionId 的，然后找到相对应的 Session 来保证用户登陆的正确性和权限的正确性， 但是在前后端分离的项目中，由于每次的 SessionId 都是不一样的，所以我这里选择的是重写 DefaultWebSessionManager 的部分方法， 然后在用户登陆的时候给前端返回 SessionId 来当用户的凭证信息，前端在请求头中携带信息，来解决 Shiro 的用户 Token 认证问题

java

package com.jg.pochi.shiro;

import com.jg.pochi.constant.CoreConstant;

import com.jg.pochi.utils.StringUtils;

import org.apache.shiro.web.session.mgt.DefaultWebSessionManager;

import org.apache.shiro.web.util.WebUtils;

import org.springframework.context.annotation.Configuration;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import java.io.Serializable;

import java.util.UUID;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 18:56

\* @Version 1.0

\*/

@Configuration

public class TokenWebSessionManager extends DefaultWebSessionManager {

@Override

protected Serializable getSessionId(ServletRequest request, ServletResponse response) {

// 从请求头获取token

String token = WebUtils.toHttp(request).getHeader(CoreConstant.TOKEN\_HEADER);

// 如果token存在，就返回token，否则就生成一个

if (StringUtils.isNotBlank(token)) {

return token;

}

return UUID.randomUUID().toString();

}

}

### 1.1.2 重写登录失效方法

shiro对于登录失效的默认处理方式是重定向到用户配置的某个页面，如果没有配置，默认重定向到 login.jsp，这对于我们前后端分离以及非jsp项目来说是不合理的，因此需要重写

java

package com.jg.pochi.shiro;

import com.alibaba.fastjson.JSON;

import com.jg.pochi.common.Result;

import com.jg.pochi.enums.ResultEnums;

import org.apache.shiro.web.filter.authc.UserFilter;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import java.io.IOException;

/\*\*

\* 重写登录失效重定向

\*

\* @Author: 杨德石

\* @Date: 2020/11/8 19:00

\* @Version 1.0

\*/

public class LoginFilter extends UserFilter {

/\*\*

\* 这个方法用于处理未登录时页面重定向的逻辑

\* 因此，只要进入了这个方法，就意味着登录失效了。

\* 我们只需要在这个方法里。给前端返回一个登陆失败的状态即可

\*

\* @param request

\* @param response

\* @throws IOException

\*/

@Override

protected void redirectToLogin(ServletRequest request, ServletResponse response) throws IOException {

// 设置响应头是json

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

// 直接写回未登录的json报文

response.getWriter().write(JSON.toJSONString(new Result<>(ResultEnums.NO\_LOGIN)));

}

}

### 重写token的保存方式

#### 知识备注：

CachingSessionDAO是一个抽象类，它有一个简单的实现EnterpriseCacheSessionDAO，一般在开发过程中推荐大家直接继承EnterpriseCacheSessionDAO。

(1)AbstractSessionDAO

提供了SessionDAO的基础实现，如生成会话ID等。

(2)CachingSessionDAO

提供了对开发者透明的会话缓存的功能，需要设置相应的CacheManager。

(3)EnterpriseCacheSessionDAO

提供了缓存功能的会话维护，默认情况下使用MapCache实现，内部使用

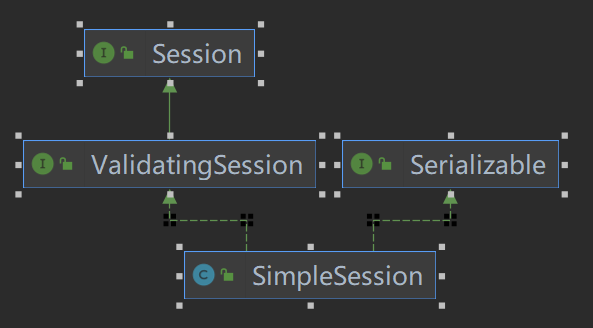
ConcurrentHashMap保存缓存的会话。

shiro默认将sessionId存在缓存中，这在单体项目中并不会有什么问题，但是后续项目如果扩展为分布式的，就存在问题了，重启服务器会导致sessionId失效，sessionId无法共享。

解决方法：

为了系统的可扩展性，这里引入redis去存储登录信息。

Java



java

package com.jg.pochi.shiro;

import org.apache.shiro.session.Session;

import org.apache.shiro.session.mgt.SimpleSession;

import org.apache.shiro.session.mgt.ValidatingSession;

import org.apache.shiro.session.mgt.eis.EnterpriseCacheSessionDAO;

import org.springframework.data.redis.core.RedisTemplate;

import org.springframework.stereotype.Component;

import javax.annotation.Resource;

import java.io.Serializable;

/\*\*

\* 重写存取sessionId的方法

\*

\* @Author: 杨德石

\* @Date: 2020/11/8 19:06

\* @Version 1.0

\*/

@Component

public class SessionDaoConfig extends EnterpriseCacheSessionDAO {

@Resource

private RedisTemplate<Serializable, Session> redisTemplate;

@Override

protected Serializable doCreate(Session session) {

// 获取sessionid

Serializable sessionId = this.generateSessionId(session);

// session要和sessionid绑定

SimpleSession simpleSession = (SimpleSession) session;

simpleSession.setId(sessionId);

return sessionId;

}

@Override

protected Session doReadSession(Serializable sessionId) {

// 从redis中读取sessionId

return redisTemplate.boundValueOps(sessionId).get();

}

@Override

protected void doUpdate(Session session) {

if (session instanceof ValidatingSession) {

ValidatingSession validatingSession = (ValidatingSession) session;

if (validatingSession.isValid()) {

redisTemplate.boundValueOps(session.getId()).set(session);

} else {

// 校验失败，说明未登录或者登录失效

redisTemplate.delete(session.getId());

}

} else {

redisTemplate.boundValueOps(session.getId()).set(session);

}

}

@Override

protected void doDelete(Session session) {

redisTemplate.delete(session.getId());

}

}

注意，如果redisTemplate使用jackson做序列化，在取出session的时候会有反序列化问题，暂未解决

### 1.1.4 编写realm

realm是shiro中用来登录授权使用的，这里先提供一个空的realm放在这，主要是把架子搭好

java

@Component("adminRealm")

public class AdminRealm extends AuthorizingRealm {

/\*\*

\* 授权方法

\*

\* @param principalCollection

\* @return

\*/

@Override

protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {

return new SimpleAuthorizationInfo();

}

/\*\*

\* 认证

\*

\* @param token

\* @return

\* @throws AuthenticationException

\*/

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {

return null;

}

}

### 1.1.5 shiro核心类

这个不需要做过多解释了，就是核心的配置，不配置的话shiro无法使用。

这里的要工作就是使上面那一堆配置全都生效。

java

package com.jg.pochi.shiro;

import org.apache.shiro.mgt.DefaultSecurityManager;

import org.apache.shiro.mgt.SecurityManager;

import org.apache.shiro.spring.LifecycleBeanPostProcessor;

import org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor;

import org.apache.shiro.spring.web.ShiroFilterFactoryBean;

import org.apache.shiro.web.mgt.DefaultWebSecurityManager;

import org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.DependsOn;

import org.springframework.data.redis.connection.RedisConnectionFactory;

import org.springframework.data.redis.core.RedisTemplate;

import org.springframework.data.redis.core.StringRedisTemplate;

import org.springframework.data.redis.serializer.JdkSerializationRedisSerializer;

import org.springframework.data.redis.serializer.StringRedisSerializer;

import javax.servlet.Filter;

import java.util.LinkedHashMap;

import java.util.Map;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 19:13

\* @Version 1.0

\*/

@Configuration

public class ShiroConfig {

@Autowired

private SysUserRealm sysUserRealm;

/\*\*

\* 配置shiroFilterFactoryBean

\*

\* @param securityManager

\* @return

\*/

@Bean("shiroFilterFactoryBean")

public ShiroFilterFactoryBean shiroFilterFactoryBean(SecurityManager securityManager) {

// 设置安全管理器

ShiroFilterFactoryBean shiroFilterFactoryBean = new ShiroFilterFactoryBean();

shiroFilterFactoryBean.setSecurityManager(securityManager);

// 把自定义的过滤器放到shiro中

Map<String, Filter> shiroFilters = shiroFilterFactoryBean.getFilters();

shiroFilters.put("authc", new LoginFilter());

// 配置需要认证或者需要放行的路径，注意，这里必须是LinkedHashMap

Map<String, String> filterMap = new LinkedHashMap<>();

filterMap.put("/\*\*", "authc");

shiroFilterFactoryBean.setFilterChainDefinitionMap(filterMap);

return shiroFilterFactoryBean;

}

/\*\*

\* 配置安全管理器

\*

\* @param sessionDaoConfig

\* @return

\*/

@Bean("securityManager")

public SecurityManager securityManager(SessionDaoConfig sessionDaoConfig) {

DefaultSecurityManager securityManager = new DefaultWebSecurityManager();

securityManager.setRealm(sysUserRealm);

// 自定义session管理，使用redis进行管理

TokenWebSessionManager sessionManager = new TokenWebSessionManager();

sessionManager.setSessionDAO(sessionDaoConfig);

securityManager.setSessionManager(sessionManager);

return securityManager;

}

/\*\*

\* 管理生命周期

\* 如果不注入可能会无法启动

\* 注意方法要是静态的

\*

\* @return

\*/

@Bean

public static LifecycleBeanPostProcessor lifecycleBeanPostProcessor() {

return new LifecycleBeanPostProcessor();

}

/\*\*

\* 使用redis存sessionId

\*

\* @param connectionFactory

\* @return

\*/

@Bean

public RedisTemplate<Object, Object> redisTemplate(RedisConnectionFactory connectionFactory) {

RedisTemplate<Object, Object> template = new RedisTemplate<>();

template.setConnectionFactory(connectionFactory);

JdkSerializationRedisSerializer serializer = new JdkSerializationRedisSerializer();

template.setValueSerializer(serializer);

template.setKeySerializer(new StringRedisSerializer());

template.afterPropertiesSet();

return template;

}

@Bean

public StringRedisTemplate stringRedisTemplate(RedisConnectionFactory factory) {

StringRedisTemplate stringRedisTemplate = new StringRedisTemplate();

stringRedisTemplate.setConnectionFactory(factory);

return stringRedisTemplate;

}

/\*\*

\* 加入注解的使用，不加入注解不生效

\*

\* @return

\*/

@Bean

@DependsOn("lifecycleBeanPostProcessor")

public static DefaultAdvisorAutoProxyCreator defaultAdvisorAutoProxyCreator() {

DefaultAdvisorAutoProxyCreator defaultAdvisorAutoProxyCreator = new DefaultAdvisorAutoProxyCreator();

defaultAdvisorAutoProxyCreator.setProxyTargetClass(true);

return defaultAdvisorAutoProxyCreator;

}

@Bean

public AuthorizationAttributeSourceAdvisor authorizationAttributeSourceAdvisor(SecurityManager securityManager) {

AuthorizationAttributeSourceAdvisor authorizationAttributeSourceAdvisor = new AuthorizationAttributeSourceAdvisor();

authorizationAttributeSourceAdvisor.setSecurityManager(securityManager);

return authorizationAttributeSourceAdvisor;

}

}

## 登录

#### 知识备注：

登录成功后只返回一个token，每个请求的请求头携带这个token

首先创建实体类 SysUser

java

package com.jg.pochi.pojo;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* 系统用户

\*

\* @Author: 杨德石

\* @Date: 2020/11/8 19:30

\* @Version 1.0

\*/

@Data

public class SysUser implements Serializable {

/\*\*

\* id

\*/

private Long id;

/\*\*

\* 用户名

\*/

private String username;

/\*\*

\* 微信的openid

\*/

private String openId;

/\*\*

\* 密码

\*/

private String password;

/\*\*

\* 邮箱

\*/

private String email;

/\*\*

\* 昵称

\*/

private String nickName;

/\*\*

\* 头像

\*/

private String header;

/\*\*

\* 备注

\*/

private String note;

/\*\*

\* 创建时间

\*/

private String createTime;

/\*\*

\* 修改时间

\*/

private String updateTime;

/\*\*

\* 最后登录时间

\*/

private String loginTime;

/\*\*

\* 账号启用状态，1是0否

\*/

private Integer status;

/\*\*

\* 是否删除，1是0否

\*/

private Integer deleted;

}

### 1.2.1 SysUserController

java

/\*\*

\* 登录

\*

\* @param sysUser

\* @return

\*/

@RequestMapping(value = "/login", method = RequestMethod.POST)

public Result<TokenVo> login(@RequestBody SysUser sysUser) {

// 校验用户名密码

if (sysUser == null || StringUtils.isBlank(sysUser.getUsername()) || StringUtils.isBlank(sysUser.getPassword())) {

return new Result<>(ResultEnums.LOGIN\_PARAM\_ERROR);

}

// 使用shiro进行登录

Subject subject = SecurityUtils.getSubject();

AuthenticationToken authenticationToken = new UsernamePasswordToken(sysUser.getUsername(), sysUser.getPassword());

try {

subject.login(authenticationToken);

} catch (Exception e) {

e.printStackTrace();

return new Result<>(ResultEnums.LOGIN\_PARAM\_ERROR);

}

// 登录成功

Serializable sessionId = subject.getSession().getId();

// 更新登录时间

sysUserService.updateLoginTime(sysUser.getUsername());

return new Result<>(new TokenVo(sessionId));

}

### 1.2.2 AdminRealm

java

package com.jg.pochi.shiro;

import com.jg.pochi.enums.ResultEnums;

import com.jg.pochi.enums.StateEnums;

import com.jg.pochi.exception.PochiException;

import com.jg.pochi.pojo.SysUser;

import com.jg.pochi.service.SysUserService;

import org.apache.shiro.authc.AuthenticationException;

import org.apache.shiro.authc.AuthenticationInfo;

import org.apache.shiro.authc.AuthenticationToken;

import org.apache.shiro.authc.SimpleAuthenticationInfo;

import org.apache.shiro.authc.UsernamePasswordToken;

import org.apache.shiro.authz.AuthorizationInfo;

import org.apache.shiro.authz.SimpleAuthorizationInfo;

import org.apache.shiro.realm.AuthorizingRealm;

import org.apache.shiro.subject.PrincipalCollection;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

/\*\*

\* 系统用户登录realm

\*

\* @author yds

\*/

@Component("sysUserRealm")

public class SysUserRealm extends AuthorizingRealm {

@Autowired

private SysUserService sysUserService;

/\*\*

\* 授权方法

\*

\* @param principalCollection

\* @return

\*/

@Override

protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {

return new SimpleAuthorizationInfo();

}

/\*\*

\* 认证

\*

\* @param token

\* @return

\* @throws AuthenticationException

\*/

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {

// 处理登录逻辑

UsernamePasswordToken usernamePasswordToken = (UsernamePasswordToken) token;

String username = usernamePasswordToken.getUsername();

SysUser sysUser = sysUserService.getByUsername(username);

if (sysUser == null) {

throw new PochiException(ResultEnums.LOGIN\_PARAM\_ERROR);

}

if (StateEnums.NOT\_ENABLE.getCode().equals(sysUser.getStatus())) {

// 未启用用户

throw new PochiException(ResultEnums.LOGIN\_PARAM\_ERROR);

}

if (StateEnums.DELETED.getCode().equals(sysUser.getDeleted())) {

// 已删除用户

throw new PochiException(ResultEnums.LOGIN\_PARAM\_ERROR);

}

return new SimpleAuthenticationInfo(sysUser, sysUser.getPassword(), this.getName());

}

}

### 1.2.3 SysUserService

java

/\*\*

\* 更新指定用户名的登录时间

\*

\* @param username

\*/

void updateLoginTime(String username);

/\*\*

\* 根据用户名查询用户

\*

\* @param username

\* @return

\*/

SysUser getByUsername(String username);

### 1.2.4 SysUserServiceImpl

java

package com.jg.pochi.service.impl;

import com.jg.pochi.mapper.SysUserMapper;

import com.jg.pochi.pojo.SysUser;

import com.jg.pochi.service.SysUserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 19:39

\* @Version 1.0

\*/

@Service

public class SysUserServiceImpl implements SysUserService {

@Autowired

private SysUserMapper sysUserMapper;

@Override

public void updateLoginTime(String username) {

sysUserMapper.updateLoginTime(username);

}

@Override

public SysUser getByUsername(String username) {

return sysUserMapper.getByUsername(username);

}

}

### 1.2.5 SysUserMapper

java

package com.jg.pochi.mapper;

import com.jg.pochi.pojo.SysUser;

import org.springframework.stereotype.Component;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 19:42

\* @Version 1.0

\*/

@Component

public interface SysUserMapper {

/\*\*

\* 更新指定用户名的登陆时间为当前时间

\*

\* @param username

\*/

void updateLoginTime(String username);

/\*\*

\* 根据用户名查询用户

\*

\* @param username

\* @return

\*/

SysUser getByUsername(String username);

}

### 1.2.6 SysUserMapper.xml

xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jg.pochi.mapper.SysUserMapper">

<resultMap id="BaseResultMap" type="com.jg.pochi.pojo.SysUser">

<id column="id" property="id"/>

<result column="username" property="username"/>

<result column="openid" property="openId"/>

<result column="password" property="password"/>

<result column="email" property="email"/>

<result column="nick\_name" property="nickName"/>

<result column="header" property="header"/>

<result column="note" property="note"/>

<result column="create\_time" property="createTime"/>

<result column="update\_time" property="updateTime"/>

<result column="login\_time" property="loginTime"/>

<result column="status" property="status"/>

<result column="deleted" property="deleted"/>

</resultMap>

<update id="updateLoginTime">

update sys\_user

set login\_time = NOW()

where username = #{username}

</update>

<select id="getByUsername" resultMap="BaseResultMap">

select id,

username,

openid,

password,

email,

nick\_name,

header,

note,

status,

deleted

from sys\_user

where username = #{username}

</select>

</mapper>

## 1.3 获取用户信息

在shiro配置文件中配置放行登录接口，其他全都需要授权

### 1.3.1 SysUserController

java

/\*\*

\* 获取登录用户

\*

\* @return

\*/

@RequestMapping(value = "/info", method = RequestMethod.GET)

public Result<SysUser> info() {

SysUser sysUser = ShiroUtils.getLoginUser();

sysUser.setPassword(null);

sysUser.setStatus(null);

sysUser.setDeleted(null);

return new Result<>(sysUser);

}

![image-20200925110404069](https://ydsmarkdown.oss-cn-beijing.aliyuncs.com/md/20200925110404.png)

## 1.4 退出登录

退出登录不能直接清除redis中的token，因为shiro中还有缓存

java

@RequestMapping(value = "/logout", method = RequestMethod.GET)

public Result<SysUser> logout() {

String token = ShiroUtils.getToken();

redisTemplate.delete(token);

return new Result<>("退出成功");

}

调用上面接口之后我们发现redis中的数据确实被清掉了，但是再次调用获取用户信息的接口发现能调通，并且成功获取到了登录用户。

因此我们需要清空shiro中的缓存

java

@RequestMapping(value = "/logout", method = RequestMethod.GET)

public Result<SysUser> logout() {

Subject currentUser = SecurityUtils.getSubject();

currentUser.logout();

return new Result<>("退出成功");

}

执行上面的接口之后，redis中的数据也被清除了。

## 1.5 前端编写

#### 知识备注：

### 子组件能通过 this.$store.dispatch 访问到

### this.$store.dispatch('changetest'xxx)

### 1.5.1 登录

启动项目，定位到登录页 login.vue

#### 页面修改

html

<div class="login-container">

<el-form ref="loginForm" :model="loginForm" :rules="loginRules" class="login-form" autocomplete="on" label-position="left">

<div class="title-container">

<h3 class="title">波奇犬屋</h3>

</div>

<el-form-item prop="username">

<span class="svg-container">

<svg-icon icon-class="user" />

</span>

<el-input

ref="username"

v-model="loginForm.username"

placeholder="Username"

name="username"

type="text"

tabindex="1"

autocomplete="on"

/>

</el-form-item>

<el-tooltip v-model="capsTooltip" content="Caps lock is On" placement="right" manual>

<el-form-item prop="password">

<span class="svg-container">

<svg-icon icon-class="password" />

</span>

<el-input

:key="passwordType"

ref="password"

v-model="loginForm.password"

:type="passwordType"

placeholder="Password"

name="password"

tabindex="2"

autocomplete="on"

@keyup.native="checkCapslock"

@blur="capsTooltip = false"

@keyup.enter.native="handleLogin"

/>

<span class="show-pwd" @click="showPwd">

<svg-icon :icon-class="passwordType === 'password' ? 'eye' : 'eye-open'" />

</span>

</el-form-item>

</el-tooltip>

<el-button :loading="loading" type="primary" style="width:100%;margin-bottom:30px;" @click.native.prevent="handleLogin">登录</el-button>

</el-form>

</div>

#### 表单校验

js

const validateUsername = (rule, value, callback) => {

if (!validUsername(value)) {

callback(new Error('请输入用户名'))

} else {

callback()

}

}

const validatePassword = (rule, value, callback) => {

if (value.length < 6) {

callback(new Error('密码长度需要在6位以上'))

} else {

callback()

}

}

#### 登录功能

追踪到 handleLogin 方法中我们发现，登录方法调用了 vuex 中的 user 模块的 login 方法，进入到里面

我们发现在这里调用了 api/user.js 中的方法，这里是接口，因此我们需要把这个文件中相关的接口都修改成我们的接口地址

javascript

import request from '@/utils/request'

const groupName = 'sysUser'

export function login(data) {

return request({

url: `/${groupName}/login`,

method: 'post',

data

})

}

export function getInfo() {

return request({

url: `/${groupName}/info`,

method: 'get'

})

}

export function logout() {

return request({

url: `/${groupName}/logout`,

method: 'get'

})

}

修改完后，就可以开始编写登录功能了

1. 对密码进行MD5加密
2. 安装 js-md5
3. sh
4. npm install --save js-md5
5. 在页面中引入
6. js
7. import md5 from 'js-md5';
8. 登录方法编写
9. js
10. login({ commit }, userInfo) {
11. const { username, password } = userInfo
12. return new Promise((resolve, reject) => {
13. login({ username: username.trim(), password: md5(password) }).then(response => {
14. const { data } = response
15. commit('SET\_TOKEN', data.token)
16. setToken(data.token)
17. resolve()
18. }).catch(error => {
19. reject(error)
20. })
21. })
22. },

### 1.5.2 获取登录信息

上面我们已经把登录功能编写完毕。 vue-element-admin 将登录和获取登录用户信息的接口拆成了两个，为了少踩坑，我们使用它的规范，现在调用这个接口。

#### 修改用户信息存储方式

我们发现在vuex中用户存储的方式不合理。

* 用户信息应该是一个对象
* 只存储了角色，没有权限

我们将用户信息改成一个对象。而roles涉及到的地方太多了，我们用roles作为权限

js

const state = {

token: getToken(),

user: {},

roles: []

}

const mutations = {

SET\_TOKEN: (state, token) => {

state.token = token

},

SET\_USER: (state, user) => {

state.user = user

},

SET\_ROLES: (state, roles) => {

state.roles = roles

}

}

修改完之后，我们需要在 getters.js 中提供对应的getter方法

js

const getters = {

sidebar: state => state.app.sidebar,

size: state => state.app.size,

device: state => state.app.device,

visitedViews: state => state.tagsView.visitedViews,

cachedViews: state => state.tagsView.cachedViews,

token: state => state.user.token,

user: state => state.user.user,

roles: state => state.user.roles,

permission\_routes: state => state.permission.routes,

errorLogs: state => state.errorLog.logs

}

export default getters

#### 修改获取信息方法

接下来修改获取用户信息的方法，在 getInfo 里。我们约定 auths 是用户的权限信息，现在权限功能还没有做，因此这里先这么写，后面再修改后端。

js

getInfo({ commit, state }) {

return new Promise((resolve, reject) => {

getInfo().then(response => {

const { data } = response

commit('SET\_USER', data)

if (data.auths) {

commit('SET\_ROLES', data.auths)

}else {

commit('SET\_ROLES', 'admin')

}

resolve(data)

}).catch(error => {

reject(error)

})

})

},

接下来，在

#### 修改路由构造方式

我们的路由最终不在前端维护，全部从后端获，用户登录后就确定了有哪些路由和菜单，因此不需要前端再根据权限去判断路由了。

找到 permission.js，这里是路由守卫，发现当没有权限的时候会去后端加载权限，并生成路由，我们把 roles 删掉

js

try {

await store.dispatch('user/getInfo')

const accessRoutes = await store.dispatch('permission/generateRoutes')

router.addRoutes(accessRoutes)

next({ ...to, replace: true })

} catch (error) {

// remove token and go to login page to re-login

await store.dispatch('user/resetToken')

Message.error(error || 'Has Error')

next(`/login?redirect=${to.path}`)

NProgress.done()

}

删掉之后，我们发现构造路由的代码是在 permission/generateRoutes 中，定位过去。

vue-element-admin 提供了两种类型的路由，constantRoutes 和 asyncRoutes。前者是写死在前端的固定路由，比如登录路由。后者是从接口获取的动态路由，我们先在前端进行操作，后面再从后端获取。

在 store/modules/permission.js 中定位到 generateRoutes 方法，移除掉对 roles 的判断

js

const actions = {

generateRoutes({ commit }) {

return new Promise(resolve => {

const accessedRoutes = filterAsyncRoutes(asyncRoutes)

commit('SET\_ROUTES', accessedRoutes)

resolve(accessedRoutes)

})

}

}

接下来修改 filterAsyncRoutes 方法，同样是移除掉权限判断

js

export function filterAsyncRoutes(routes) {

const res = []

routes.forEach(route => {

const tmp = { ...route }

if (tmp.children) {

tmp.children = filterAsyncRoutes(tmp.children)

}

res.push(tmp)

})

return res

}

而 hasPermission 方法则可以直接删除了

接下来再测试登录功能，发现已经正常。

登录后我们发现报错了，定位到 PanThumb 发现这个组件在 BoxCard 被引用。原来是我们之前删掉了 vuex 中的 avatar，这是用户头像，因此报错了，我们只需要修改成正常的用户头像即可。

js、

computed: {

...mapGetters([

'user',

'roles'

])

}

html

<pan-thumb :image="user.header" class="panThumb" />

而在 layout/Navbar.vue 中也存在同样的问题

html

<img :src="user.header" class="user-avatar">

js

computed: {

...mapGetters([

'sidebar',

'user',

'device'

])

},

### 1.5.3 退出登录

退出登录的实现非常简单，调用退出登录接口然后清空token即可，修改 logout 方法

js

logout({ commit, state, dispatch }) {

return new Promise((resolve, reject) => {

logout().then(() => {

commit('SET\_TOKEN', '')

commit('SET\_ROLES', [])

removeToken()

resetRouter()

dispatch('tagsView/delAllViews', null, { root: true })

resolve()

}).catch(error => {

reject(error)

})

})

},

# 2. 用户管理

## 2.1 后端

### 2.1.1 SysUserController

#### 知识备注：

1、@Transactional*(*rollbackFor = Exception.class*)*

在@Transactional注解中如果不配置rollbackFor属性,那么事物只会在遇到RuntimeException的时候才会回滚,加上rollbackFor=Exception.class,可以让事物在遇到非运行时异常时也回滚。

由于Spring管理事务默认回滚只针对RunTimeException和UnCheckedException，如果让CheckedException也回滚就需要加上这个。

1 让checked例外也回滚：在整个方法前加上 @Transactional(rollbackFor=Exception.class)

2 让unchecked例外不回滚： @Transactional(notRollbackFor=RunTimeException.class)

3 不需要事务管理的(只查询的)方法：@Transactional(propagation=Propagation.NOT\_SUPPORTED)

2、resultMap="BaseResultMap"

关于mybatis的resultMap="BaseResultMap"说明

mybatis 非常的智能，如果配置了resultMap，返回值统一使用 resultMap=“BaseResultMap”，mybatis会根据查询到的条目数量自动进行判断，如果是一条就返回对象，如果是多条就返回List对象列表。

java

/\*\*

\* 保存用户

\*

\* @param sysUser

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<?> save(@RequestBody SysUser sysUser) {

// 参数校验

if (StringUtils.isBlank(sysUser.getUsername())) {

return new Result<>(ResultEnums.PARAMS\_NULL, "用户名不能为空！");

}

if (StringUtils.isBlank(sysUser.getPassword())) {

return new Result<>(ResultEnums.PARAMS\_NULL, "密码不能为空");

}

sysUserService.save(sysUser);

return new Result<>("添加成功");

}

/\*\*

\* 修改用户

\* 修改接口一般不提供密码修改功能

\*

\* @param sysUser

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<?> update(@RequestBody SysUser sysUser) {

sysUserService.update(sysUser);

return new Result<>("修改成功");

}

/\*\*

\* 删除接口

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/delete/{id}", method = RequestMethod.DELETE)

public Result<?> delete(@PathVariable Long id) {

sysUserService.delete(id);

return new Result<>("删除成功");

}

/\*\*

\* 启用用户

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/enable/{id}", method = RequestMethod.PUT)

public Result<?> enable(@PathVariable Long id) {

sysUserService.enable(id);

return new Result<>("启用成功");

}

/\*\*

\* 封禁用户

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/disable/{id}", method = RequestMethod.PUT)

public Result<?> disable(@PathVariable Long id) {

sysUserService.disable(id);

return new Result<>("禁用成功");

}

/\*\*

\* 分页查询

\*

\* @param page

\* @return

\*/

@RequestMapping(value = "/getByPage", method = RequestMethod.POST)

public Result<Page<SysUser>> getByPage(@RequestBody Page<SysUser> page) {

page = sysUserService.getByPage(page);

return new Result<>(page);

}

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysUser> get(@PathVariable Long id) {

SysUser sysUser = sysUserService.get(id);

return new Result<>(sysUser);

}

### 2.1.2 SysUserService

java

/\*\*

\* 保存用户

\* @param sysUser

\*/

void save(SysUser sysUser);

/\*\*

\* 修改用户

\* @param sysUser

\*/

void update(SysUser sysUser);

/\*\*

\* 删除用户

\* @param id

\*/

void delete(Long id);

/\*\*

\* 启用

\* @param id

\*/

void enable(Long id);

/\*\*

\* 禁用

\* @param id

\*/

void disable(Long id);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

Page<SysUser> getByPage(Page<SysUser> page);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysUser get(Long id);

### 2.1.3 SysUserServiceImpl

java

@Override

public void save(SysUser sysUser) {

// id用雪花算法生成

sysUser.setId(idWorker.nextId());

sysUserMapper.save(sysUser);

}

@Override

public void update(SysUser sysUser) {

sysUserMapper.update(sysUser);

}

@Override

public void delete(Long id) {

sysUserMapper.delete(id);

}

@Override

@Transactional(rollbackFor = Exception.class)

public void enable(Long id) {

// 先查再改

SysUser sysUser = sysUserMapper.getById(id);

sysUser.setStatus(StateEnums.ENABLED.getCode());

sysUserMapper.updateStatus(sysUser);

}

@Override

@Transactional(rollbackFor = Exception.class)

public void disable(Long id) {

// 先查再改

SysUser sysUser = sysUserMapper.getById(id);

sysUser.setStatus(StateEnums.NOT\_ENABLE.getCode());

sysUserMapper.updateStatus(sysUser);

}

@Override

public Page<SysUser> getByPage(Page<SysUser> page) {

// 设置默认的当前页和每页条数

Integer pageNumber = page.getPageNumber();

if (pageNumber == null || pageNumber < 1) {

pageNumber = 1;

page.setPageNumber(pageNumber);

}

List<SysUser> userList = sysUserMapper.getByPage(page);

Integer totalCount = sysUserMapper.countByPage(page);

page.setList(userList);

page.setTotalCount(totalCount);

return page;

}

@Override

public SysUser get(Long id) {

return sysUserMapper.getById(id);

}

### 2.1.4 SysUserMapper

java

/\*\*

\* 添加用户

\*

\* @param sysUser

\*/

void save(SysUser sysUser);

/\*\*

\* 修改用户

\*

\* @param sysUser

\*/

void update(SysUser sysUser);

/\*\*

\* 删除用户

\* 逻辑删除

\*

\* @param id

\*/

void delete(Long id);

/\*\*

\* 更新状态值

\*

\* @param sysUser

\*/

void updateStatus(SysUser sysUser);

/\*\*

\* 根据id查询

\*

\* @param id

\* @return

\*/

SysUser getById(Long id);

/\*\*

\* 根据page分页查询

\*

\* @param page

\* @return

\*/

List<SysUser> getByPage(Page<SysUser> page);

/\*\*

\* 根据page查询总数

\*

\* @param page

\* @return

\*/

Integer countByPage(Page<SysUser> page);

### 2.1.5 SysUserMapper.xml

xml

<insert id="save">

insert into sys\_user(id, username, openid, password,

email, nick\_name,

header, note)

values (#{id}, #{username}, #{openId}, #{password},

#{email}, #{nickName}, #{header}, #{note})

</insert>

<update id="updateLoginTime">

update sys\_user

set login\_time = NOW()

where username = #{username}

</update>

<update id="update">

update sys\_user

set openid = #{openId},

email = #{email},

nick\_name = #{nickName},

header = #{header},

note = #{note}

where id = #{id}

</update>

<update id="delete">

update sys\_user

set deleted = 1

where id = #{id}

</update>

<update id="updateStatus">

update sys\_user

set status = #{status}

where id = #{id}

</update>

<select id="getByUsername" resultMap="BaseResultMap">

select id,

username,

openid,

password,

email,

nick\_name,

header,

note,

status,

deleted

from sys\_user

where username = #{username}

</select>

<select id="getById" resultMap="BaseResultMap">

select id,

username,

openid,

password,

email,

nick\_name,

header,

note,

create\_time,

update\_time,

login\_time,

status

from sys\_user

where id = #{id}

</select>

<select id="getByPage" resultMap="BaseResultMap">

select id,

username,

nick\_name,

header,

note,

create\_time,

username,

login\_time,

status

from sys\_user

where deleted = 0

<if test="params.username != null and params.username!=''">

and username = #{params.username}

</if>

<if test="params.nickName != null and params.nickName!=''">

and nick\_name like concat('%',#{params.nickName},'%')

</if>

<if test="params.status != null and params.status!=''">

and status = #{params.status}

</if>

limit #{index}, #{pageSize}

</select>

<select id="countByPage" resultType="java.lang.Integer">

select

count(\*)

from sys\_user

where deleted = 0

<if test="params.username != null and params.username!=''">

and username = #{params.username}

</if>

<if test="params.nickName != null and params.nickName!=''">

and nick\_name like concat('%',#{params.nickName},'%')

</if>

<if test="params.status != null and params.status!=''">

and status = #{params.status}

</if>

</select>

## 2.2 前端

### 2.2.1 API定义

在 api 目录下创建 sysUser.js

js

import request from '@/utils/request'

const groupName = 'sysUser'

export default {

/\*\*

\* 添加

\*/

save(sysUser) {

return request({

url: `/${groupName}/save`,

method: 'post',

data: sysUser

})

},

/\*\*

\* 修改

\*/

update(sysUser) {

return request({

url: `/${groupName}/update`,

method: 'put',

data: sysUser

})

},

/\*\*

\* 分页

\*/

getByPage(page) {

return request({

url: `/${groupName}/getByPage`,

method: 'post',

data: page

})

},

/\*\*

\* 启用

\*/

enableById(id) {

return request({

url: `/${groupName}/enable/${id}`,

method: 'put'

})

},

/\*\*

\* 禁用

\*/

disableById(id) {

return request({

url: `/${groupName}/disable/${id}`,

method: 'put'

})

},

/\*\*

\* 删除

\*/

deleteById(id) {

return request({

url: `/${groupName}/delete/${id}`,

method: 'delete'

})

},

/\*\*

\* 根据id查询

\*/

get(id) {

return request({

url: `/${groupName}/get/${id}`,

method: 'get'

})

}

}

### 2.2.2 路由配置

由于现阶段没有动态路由，因此先配死，找到 router/index.js 文件，在 asyncRoutes 下配置路由

js

{

path: '/sys',

component: Layout,

redirect: '/sys/user',

alwaysShow: true, // will always show the root menu

name: 'system',

meta: {

title: '系统管理',

icon: 'lock'

},

children: [

{

path: 'user',

component: () => import('@/views/system/user/sys-user-list'),

name: 'user',

meta: {

title: '用户管理',

icon: 'lock'

}

}

]

},

### 2.2.3 页面编写

在 views 下新建目录 system/user ，目录下分别创建 sys-user-list.vue、sys-user-info.vue、sys-user-add.vue、sys-user-update.vue 文件。

#### 知识备注：

*低级错误：*POST写成PSOT。。。。。



1. <template slot-scope="xxxx">是一个固定写法

通过 Scoped slot 可以获取到 row, column, $index 和 store（table 内部的状态管理）的数据

例如：

          <template slot-scope="{row}">

            <el-image

              style="width: 50px; height: 50px"

              :src="row.header"

              fit="fill"

              :preview-src-list="[row.header]"

            />

背景:

vue2.5版本中slot-scope取代了scope，来实现作用域插槽，主要用在组件调用中，具体在template标签上面使用slot-scope来获取插槽slot上面的属性值，获取值的为一个对象，slot-scope=”它可以取任意字符串”，在element-ui的组件中经常看到。

1. <style> scoped 属性，则样式仅仅应用到 style 元素的父元素及其子元素。

#### sys-user-list

vue

<template>

<div>

<!-- 搜索栏开始 -->

<div class="search-form">

<el-form :inline="true" :model="page.params" size="small">

<el-form-item>

<el-input v-model="page.params.username" clearable placeholder="请输入用户名" />

</el-form-item>

<el-form-item>

<el-input v-model="page.params.nickName" clearable placeholder="请输入昵称" />

</el-form-item>

<el-form-item>

<el-select v-model="page.params.status" clearable placeholder="请选择启用状态">

<el-option label="启用" :value="1" />

<el-option label="禁用" :value="0" />

</el-select>

</el-form-item>

<el-form-item>

<el-button type="primary" icon="el-icon-search" @click="search">查询</el-button>

<el-button type="warning" icon="el-icon-refresh" @click="page.params = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

<!-- 搜索栏结束 -->

<!-- 操作按钮组 -->

<div class="button-group">

<el-button type="primary" size="small" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

<!-- 操作按钮组结束 -->

<!-- 数据表格开始 -->

<div class="data-table">

<el-table header-row-class-name="pochi-table-header" :data="dataPage.list" stripe style="width: 100%">

<el-table-column prop="username" label="账号" />

<el-table-column prop="nickName" label="昵称" />

<el-table-column prop="email" label="邮箱" />

<el-table-column prop="header" label="头像">

<template slot-scope="{row}">

<el-image

style="width: 50px; height: 50px"

:src="row.header"

fit="fill"

:preview-src-list="[row.header]"

/>

</template>

</el-table-column>

<el-table-column prop="createTime" label="创建时间" />

<el-table-column prop="updateTime" label="更新时间" />

<el-table-column prop="loginTime" label="登录时间" />

<el-table-column prop="status" label="启用状态">

<template slot-scope="{row}">

<el-switch

v-model="row.status"

:active-value="1"

:inactive-value="0"

@change="changeStatus(row)"

/>

</template>

</el-table-column>

<el-table-column prop="note" label="备注" />

<el-table-column label="操作">

<template slot-scope="{row}">

<el-button type="text" icon="el-icon-document" @click="toInfo(row.id)">详情</el-button>

<el-dropdown class="handle-button">

<span class="el-dropdown-link">

操作<i class="el-icon-arrow-down el-icon--right" />

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item>

<el-button type="text" icon="el-icon-edit" @click="toUpdate(row.id)">修改</el-button>

</el-dropdown-item>

<el-dropdown-item @click.native="deleteById(row.id)">

<el-button type="text" icon="el-icon-delete">删除</el-button>

</el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

</div>

<!-- 分页组件开始 -->

<div class="pageable">

<el-pagination

:current-page="page.pageNumber"

:page-sizes="[10, 20, 30, 50]"

:page-size="10"

background

layout="total, sizes, prev, pager, next, jumper"

:total="dataPage.totalCount"

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

/>

</div>

<!-- 分页组件结束 -->

<!-- 添加用户弹窗 -->

<el-dialog

title="添加用户"

:visible.sync="addDialog"

width="45%"

>

<sysUserAdd @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 添加用户弹窗结束 -->

<!-- 修改弹窗 -->

<el-dialog

title="修改用户"

:visible.sync="updateDialog"

width="45%"

>

<sysUserUpdate :active-id="activeId" @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 修改弹窗结束 -->

<!-- 详情弹窗开始 -->

<el-dialog

title="用户详情"

:visible.sync="infoDialog"

width="30%"

>

<sysUserInfo :active-id="activeId" />

</el-dialog>

<!-- 详情弹窗结束 -->

</div>

</template>

<script>

import sysUserApi from '@/api/sys-user'

import sysUserAdd from './sys-user-add'

import sysUserUpdate from './sys-user-update'

import sysUserInfo from './sys-user-info'

export default {

components: {

sysUserAdd,

sysUserUpdate,

sysUserInfo

},

data() {

return {

// 分页对象

page: {

// 分页传参

params: {},

pageNumber: 1,

pageSize: 10

},

// 当前点击的用户

activeId: '',

// 控制添加用户弹窗展示

addDialog: false,

// 控制修改用户弹窗展示

updateDialog: false,

// 控制用户详情弹窗展示

infoDialog: false,

// 数据 page 对象

dataPage: {}

}

},

created() {

this.getByPage()

},

methods: {

// 搜索

search() {

// 搜索时，要还原为第一页

this.page.pageNumber = 1

this.getByPage()

},

// 每页条数变更时触发

handleSizeChange(val) {

this.page.pageSize = val

this.getByPage()

},

// 当前页变更时触发

handleCurrentChange(val) {

this.page.pageNumber = val

this.getByPage()

},

// 弹出添加弹窗

toAdd() {

this.addDialog = true

},

// 打开修改弹窗

toUpdate(id) {

this.activeId = id

this.updateDialog = true

},

// 查看用户详情

toInfo(id) {

this.activeId = id

this.infoDialog = true

},

// 分页查询

getByPage() {

sysUserApi.getByPage(this.page).then(res => {

this.dataPage = res.data

})

},

// 关闭弹窗

closeDialog() {

this.addDialog = false

this.updateDialog = false

this.infoDialog = false

},

// 根据id删除

deleteById(id) {

this.$confirm('是否删除该用户?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'error'

}).then(() => {

sysUserApi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

})

},

// 改变页面开关触发

changeStatus(row) {

const status = row.status

if (status === 0) {

// 禁用

this.$confirm('是否禁用该用户，禁用后将无法登陆?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

sysUserApi.disableById(row.id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

row.status = 1

})

} else {

// 启用

this.$confirm('是否启用该用户?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'success'

}).then(() => {

sysUserApi.enableById(row.id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

row.status = 0

})

}

}

}

}

</script>

<style scoped>

.button-group {

margin-bottom: 15px;

}

.pageable {

margin-top: 15px;

}

</style>

<style>

.pochi-table-header th {

background: #DFDFDF;

color: #131111;

}

</style>

#### sys-user-info

vue

<template>

<div>

<el-form ref="form" :model="sysUser" label-width="80px" size="small">

<el-row>

<el-col :span="24" class="header-container">

<el-image

style="width: 100px; height: 100px"

:src="sysUser.header"

fit="fill"

/>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8">

<el-form-item label="账号">

{{ sysUser.username }}

</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="角色">

{{ sysUser.role }}

</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="昵称">

{{ sysUser.nickName }}

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="邮箱">

{{ sysUser.email }}

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="启用状态">

{{ sysUser.status === 1 ? '启用': '未启用' }}

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="创建时间">

{{ sysUser.createTime }}

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="更新时间">

{{ sysUser.updateTime }}

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="最后登录">

{{ sysUser.loginTime }}

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="备注">

{{ sysUser.note }}

</el-form-item>

</el-col>

</el-row>

</el-form>

</div>

</template>

<script>

import sysUserApi from '@/api/sys-user'

export default {

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

// 回显用户信息对象

sysUser: {}

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

methods: {

// 根据id查询

getById(id) {

sysUserApi.get(id).then(res => {

this.sysUser = res.data

})

}

}

}

</script>

<style scoped>

.header-container {

text-align: center;

margin-bottom: 15px;

}

</style>

#### sys-user-add

vue

<template>

<div>

<el-form ref="form" :model="sysUser" :rules="rules" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="账号" prop="username">

<el-input v-model="sysUser.username" placeholder="请输入账号" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="密码" prop="password">

<el-input v-model="sysUser.password" show-password placeholder="请输入密码" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8">

<el-form-item label="昵称" prop="nickName">

<el-input v-model="sysUser.nickName" placeholder="请输入昵称" clearable />

</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="角色">

<el-input v-model="sysUser.role" placeholder="请输入角色" clearable />

</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="邮箱">

<el-input v-model="sysUser.email" placeholder="请输入邮箱" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="备注">

<el-input v-model="sysUser.note" :rows="2" type="textarea" placeholder="请输入备注" clearable />

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="sysUser = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysUserApi from '@/api/sys-user'

import md5 from 'js-md5'

export default {

data() {

return {

// 用户表单

sysUser: {},

// 表单校验对象

rules: {

username: [

{ required: true, message: '请输入用户名', trigger: 'blur' },

{ min: 5, message: '用户名最短5位', trigger: 'change' }

],

password: [

{ required: true, message: '请输入密码', trigger: 'blur' },

{ min: 5, message: '密码最短5位', trigger: 'change' }

],

nickName: [

{ required: true, message: '请输入昵称', trigger: 'blur' }

]

}

}

},

methods: {

// 添加

add() {

// 1.密码要md5加密

const form = { ...this.sysUser }

form.password = md5(form.password)

// 2.表单校验

sysUserApi.save(form).then(res => {

this.$message.success(res.msg)

// 关闭对话框，并刷新列表

this.$emit('close')

this.$emit('after')

})

}

}

}

</script>

<style>

</style>

#### sys-user-update

vue

<template>

<div>

<el-form ref="form" :model="sysUser" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="账号">

<el-input v-model="sysUser.username" readonly placeholder="请输入账号" />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="角色">

<el-input v-model="sysUser.role" placeholder="请输入角色" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="昵称">

<el-input v-model="sysUser.nickName" placeholder="请输入昵称" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="邮箱">

<el-input v-model="sysUser.email" placeholder="请输入邮箱" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="备注">

<el-input v-model="sysUser.note" :rows="2" type="textarea" placeholder="请输入备注" clearable />

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="update">修改</el-button>

<el-button type="warning" @click="sysUser = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysUserApi from '@/api/sys-user'

export default {

props: {

// 调用方传来的用户ID

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单数据

sysUser: {}

}

},

watch: {

activeId: {

// 该属性作用就是让监听器立即生效

immediate: true,

// 处理监听器的实际逻辑

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

methods: {

// 根据id查询用户

getById(id) {

sysUserApi.get(id).then(res => {

this.sysUser = res.data

})

},

// 修改用户

update() {

sysUserApi.update(this.sysUser).then(res => {

this.$message.success(res.msg)

// 关闭对话框，并刷新列表

this.$emit('close')

this.$emit('after')

})

}

}

}

</script>

<style>

</style>

### 2.2.4 表单校验

#### 知识备注：

给组件绑定原生事件@click.native

ElementUI提供了很强大的表单校验功能，只需要给表单提供一个 rule 就可以指定校验方式。这里以添加页为例。

trigger：什么时候展示，'blur'是失去焦点时展示

vue

rules: {

username: [

{ required: true, message: '请输入用户名', trigger: 'blur' },

{ min: 5, message: '用户名最短5位', trigger: 'change' }

],

password: [

{ required: true, message: '请输入密码', trigger: 'blur' },

{ min: 5, message: '密码最短5位', trigger: 'change' }

],

nickName: [

{ required: true, message: '请输入昵称', trigger: 'blur' }

]

}

# 3. 角色管理

这里我们先开发添加、修改、删除、更新功能

## 3.1 后端

### 3.1.1 SysRole

java

package com.jg.pochi.pojo;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* 角色

\* @Author: 杨德石

\* @Date: 2020/11/12 0:26

\* @Version 1.0

\*/

@Data

public class SysRole implements Serializable {

/\*\*

\* 角色ID，自增

\*/

private Long roleId;

/\*\*

\* 角色名

\*/

private String roleName;

/\*\*

\* 排序值，越小越靠前

\*/

private Integer roleSort;

/\*\*

\* 创建人

\*/

private String createBy;

/\*\*

\* 创建时间

\*/

private String createTime;

/\*\*

\* 更新人

\*/

private String updateBy;

/\*\*

\* 更新时间

\*/

private String updateTime;

/\*\*

\* 逻辑删除，1是0否

\*/

private Integer delete;

}

### 3.1.2 SysRoleController

java

package com.jg.pochi.controller;

import com.jg.pochi.common.Page;

import com.jg.pochi.common.Result;

import com.jg.pochi.pojo.SysRole;

import com.jg.pochi.service.SysRoleService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/12 0:31

\* @Version 1.0

\*/

@RestController

@RequestMapping("/sysRole")

public class SysRoleController {

@Autowired

private SysRoleService sysRoleService;

/\*\*

\* 添加

\* @param sysRole

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<?> save(@RequestBody SysRole sysRole) {

sysRoleService.save(sysRole);

return new Result<>("添加成功");

}

/\*\*

\* 修改

\* @param sysRole

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<?> update(@RequestBody SysRole sysRole) {

sysRoleService.update(sysRole);

return new Result<>("修改成功");

}

/\*\*

\* 根据id删除

\* @param id

\* @return

\*/

@RequestMapping(value = "/delete/{id}", method = RequestMethod.DELETE)

public Result<?> delete(@PathVariable Long id) {

sysRoleService.delete(id);

return new Result<>("删除成功");

}

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysRole> get(@PathVariable Long id) {

SysRole sysRole = sysRoleService.get(id);

return new Result<>(sysRole);

}

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

@RequestMapping(value = "/getByPage", method = RequestMethod.POST)

public Result<Page<SysRole>> getByPage(@RequestBody Page<SysRole> page) {

page = sysRoleService.getByPage(page);

return new Result<>(page);

}

}

### 3.1.3 SysRoleService

java

package com.jg.pochi.service;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysRole;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/12 0:30

\* @Version 1.0

\*/

public interface SysRoleService {

/\*\*

\* 添加

\* @param sysRole

\*/

void save(SysRole sysRole);

/\*\*

\* 修改

\* @param sysRole

\*/

void update(SysRole sysRole);

/\*\*

\* 根据id删除

\* @param id

\*/

void delete(Long id);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysRole get(Long id);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

Page<SysRole> getByPage(Page<SysRole> page);

}

### 3.1.4 SysRoleServiceImpl

java

package com.jg.pochi.service.impl;

import com.jg.pochi.common.Page;

import com.jg.pochi.enums.StateEnums;

import com.jg.pochi.mapper.SysUserMapper;

import com.jg.pochi.pojo.SysUser;

import com.jg.pochi.service.SysUserService;

import com.jg.pochi.utils.IdWorker;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 19:39

\* @Version 1.0

\*/

@Service

public class SysUserServiceImpl implements SysUserService {

@Autowired

private SysUserMapper sysUserMapper;

@Autowired

private IdWorker idWorker;

@Override

public void updateLoginTime(String username) {

sysUserMapper.updateLoginTime(username);

}

@Override

public SysUser getByUsername(String username) {

return sysUserMapper.getByUsername(username);

}

@Override

public void save(SysUser sysUser) {

// id用雪花算法生成

sysUser.setId(idWorker.nextId());

sysUserMapper.save(sysUser);

}

@Override

public void update(SysUser sysUser) {

sysUserMapper.update(sysUser);

}

@Override

public void delete(Long id) {

sysUserMapper.delete(id);

}

@Override

@Transactional(rollbackFor = Exception.class)

public void enable(Long id) {

// 先查再改

SysUser sysUser = sysUserMapper.getById(id);

sysUser.setStatus(StateEnums.ENABLED.getCode());

sysUserMapper.updateStatus(sysUser);

}

@Override

@Transactional(rollbackFor = Exception.class)

public void disable(Long id) {

// 先查再改

SysUser sysUser = sysUserMapper.getById(id);

sysUser.setStatus(StateEnums.NOT\_ENABLE.getCode());

sysUserMapper.updateStatus(sysUser);

}

@Override

public Page<SysUser> getByPage(Page<SysUser> page) {

// 设置默认的当前页和每页条数

Integer pageNumber = page.getPageNumber();

if (pageNumber == null || pageNumber < 1) {

pageNumber = 1;

page.setPageNumber(pageNumber);

}

List<SysUser> userList = sysUserMapper.getByPage(page);

Integer totalCount = sysUserMapper.countByPage(page);

page.setList(userList);

page.setTotalCount(totalCount);

return page;

}

@Override

public SysUser get(Long id) {

return sysUserMapper.getById(id);

}

}

### 3.1.5 SysRoleMapper

java

package com.jg.pochi.mapper;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysRole;

import org.springframework.stereotype.Component;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/12 0:29

\* @Version 1.0

\*/

@Component

public interface SysRoleMapper {

/\*\*

\* 保存

\* @param sysRole

\*/

void save(SysRole sysRole);

/\*\*

\* 更新

\* @param sysRole

\*/

void update(SysRole sysRole);

/\*\*

\* 根据id删除

\* @param id

\*/

void delete(Long id);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysRole get(Long id);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

List<SysRole> getByPage(Page<SysRole> page);

/\*\*

\* 查询总数

\* @param page

\* @return

\*/

Integer countByPage(Page<SysRole> page);

}

### 3.1.6 SysRoleMapper.xml

xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jg.pochi.mapper.SysRoleMapper">

<resultMap id="BaseResultMap" type="com.jg.pochi.pojo.SysRole">

<id column="role\_id" property="roleId"/>

<result column="role\_name" property="roleName"/>

<result column="role\_sort" property="roleSort"/>

<result column="create\_by" property="createBy"/>

<result column="create\_time" property="createTime"/>

<result column="update\_by" property="updateBy"/>

<result column="update\_time" property="updateTime"/>

<result column="delete" property="delete"/>

</resultMap>

<insert id="save">

insert into sys\_role(role\_name, role\_sort, create\_by, update\_by)

VALUES (#{roleName}, #{roleSort}, #{createBy}, #{updateBy})

</insert>

<update id="update">

update sys\_role

set role\_name = #{roleName},

role\_sort = #{roleSort},

update\_by = #{updateBy}

where role\_id = #{roleId}

</update>

<update id="delete">

update sys\_role

set deleted = 1

where role\_id = #{roleId}

</update>

<select id="get" resultMap="BaseResultMap">

select role\_id,

role\_name,

role\_sort,

create\_by,

create\_time,

update\_by,

update\_time

from sys\_role

where role\_id = #{roleId}

</select>

<select id="getByPage" resultMap="BaseResultMap">

select role\_id,

role\_name,

role\_sort,

create\_by,

create\_time,

update\_by,

update\_time

from sys\_role

where deleted = 0

<if test="params.roleName != null and params.roleName != ''">

and role\_name = #{params.roleName}

</if>

order by role\_sort asc

limit #{index}, #{pageSize}

</select>

<select id="countByPage" resultType="java.lang.Integer">

select count(\*)

from sys\_role

where deleted = 0

<if test="params.roleName != null and params.roleName != ''">

and role\_name = #{params.roleName}

</if>

</select>

</mapper>

## 3.2 前端

### 3.2.1 API定义

在 api 目录下创建 sysRole.js 文件

js

import request from '@/utils/request'

const groupName = 'sysRole'

export default {

/\*\*

\* 添加

\*/

save(sysUser) {

return request({

url: `/${groupName}/save`,

method: 'post',

data: sysUser

})

},

/\*\*

\* 修改

\*/

update(sysUser) {

return request({

url: `/${groupName}/update`,

method: 'put',

data: sysUser

})

},

/\*\*

\* 分页

\*/

getByPage(page) {

return request({

url: `/${groupName}/getByPage`,

method: 'post',

data: page

})

},

/\*\*

\* 删除

\*/

deleteById(id) {

return request({

url: `/${groupName}/delete/${id}`,

method: 'delete'

})

},

/\*\*

\* 根据id查询

\*/

get(id) {

return request({

url: `/${groupName}/get/${id}`,

method: 'get'

})

}

}

### 3.2.2 路由配置

接着在系统管理下面配置一个角色管理菜单

js

<template>

<div>

<!-- 搜索表单 -->

<div class="search-form">

<el-form ref="form" :model="page.params" label-width="80px" :inline="true" size="small">

<el-form-item>

<el-input v-model="page.params.roleName" placeholder="角色名" clearable />

</el-form-item>

<el-form-item>

<el-button type="primary" icon="el-icon-search" @click="search">搜索</el-button>

<el-button type="warning" icon="el-icon-refresh" @click="page.params = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

<!-- 搜索表单结束 -->

<!-- 按钮组 -->

<div class="button-group">

<el-button type="primary" size="small" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

<!-- 按钮组结束 -->

<!-- 表格开始 -->

<div class="data-table">

<el-table

header-row-class-name="pochi-table-header"

:data="dataPage.list"

stripe

style="width: 100%"

>

<el-table-column prop="roleId" label="角色编号" />

<el-table-column prop="roleName" label="角色名" />

<el-table-column prop="roleSort" label="排序" />

<el-table-column prop="createBy" label="创建人" />

<el-table-column prop="createTime" label="创建时间" />

<el-table-column prop="updateBy" label="更新人" />

<el-table-column prop="updateTime" label="更新时间" />

<el-table-column label="操作">

<template slot-scope="{row}">

<el-dropdown class="handle-button">

<span class="el-dropdown-link">

操作<i class="el-icon-arrow-down el-icon--right" />

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item>

<el-button type="text" icon="el-icon-edit" @click="toUpdate(row.roleId)">修改</el-button>

</el-dropdown-item>

<el-dropdown-item @click.native="deleteById(row.roleId)">

<el-button type="text" icon="el-icon-delete">删除</el-button>

</el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

</div>

<!-- 表格结束 -->

<!-- 分页组件开始 -->

<div class="pageable">

<el-pagination

:current-page="page.pageNumber"

:page-sizes="[10, 20, 30, 50]"

:page-size="10"

background

layout="total, sizes, prev, pager, next, jumper"

:total="dataPage.totalCount"

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

/>

</div>

<!-- 分页组件结束 -->

<!-- 添加弹窗 -->

<el-dialog title="添加角色" :visible.sync="addDialog" width="30%">

<sysRoleAdd @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 添加弹窗结束 -->

<!-- 修改弹窗 -->

<el-dialog title="修改角色" :visible.sync="updateDialog" width="30%">

<sysRoleUpdate :active-id="activeId" @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 修改弹窗结束 -->

</div>

</template>

<script>

import sysRoleAdd from './sys-role-add'

import sysRoleUpdate from './sys-role-update'

import sysRoleApi from '@/api/sys-role'

export default {

components: {

sysRoleAdd,

sysRoleUpdate

},

data() {

return {

// 分页对象

page: {

// 搜索条件

params: {},

// 当前页

pageNumber: 1,

// 每页条数

pageSize: 10

},

// 当前点击的角色ID

activeId: '',

// 数据显示分页对象

dataPage: {},

// 控制添加弹窗展示

addDialog: false,

// 控制修改弹窗展示

updateDialog: false

}

},

created() {

this.getByPage()

},

methods: {

// 搜索

search() {

// 页数回归第一页

this.page.pageNumber = 1

this.getByPage()

},

// 打开添加弹窗

toAdd() {

this.addDialog = true

},

// 打开修改弹窗

toUpdate(id) {

this.activeId = id

this.updateDialog = true

},

// 根据id删除

deleteById(id) {

this.$confirm('是否删除该角色?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'error'

}).then(() => {

sysRoleApi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

})

},

// 分页查询

getByPage() {

sysRoleApi.getByPage(this.page).then(res => {

this.dataPage = res.data

})

},

// 每页条数发生改变

handleSizeChange(val) {

this.page.pageSize = val

this.getByPage()

},

// 关闭弹窗

closeDialog() {

this.addDialog = false

this.updateDialog = false

},

// 当前页发生改变

handleCurrentChange(val) {

this.page.pageNumber = val

this.getByPage()

}

}

}

</script>

<style>

</style>

### 3.2.3 页面编写

#### sys-role-list

vue

<template>

<div>

<!-- 搜索表单开始 -->

<div class="search-form">

<el-form :inline="true" size="mini" :model="page.params">

<el-form-item>

<el-input v-model="page.params.eq\_roleName" placeholder="角色名" />

</el-form-item>

<el-form-item>

<el-button type="primary" icon="el-icon-search" @click="search">查询</el-button>

<el-button type="warning" icon="el-icon-refresh" @click="page.params = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

<!-- 搜索表单结束 -->

<!-- 菜单组开始 -->

<div class="mid-button-group">

<el-button size="mini" type="primary" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

<!-- 菜单组结束 -->

<!-- 数据表格开始 -->

<div class="data-table">

<el-table

v-loading="loading"

:data="dataPage.list"

header-row-class-name="table-header"

stripe

style="width: 100%"

>

<el-table-column

prop="roleId"

label="角色编号"

width="200"

/>

<el-table-column

prop="roleName"

label="角色名"

width="200"

/>

<el-table-column

prop="roleSort"

width="200"

label="排序"

/>

<el-table-column

prop="createBy"

label="创建人"

/>

<el-table-column

prop="createTime"

label="创建时间"

/>

<el-table-column

prop="updateBy"

label="更新人"

/>

<el-table-column

prop="updateTime"

label="更新时间"

/>

<el-table-column

label="操作"

>

<template slot-scope="{row}">

<el-dropdown class="button-group">

<span class="el-dropdown-link">

操作<i class="el-icon-arrow-down el-icon--right" />

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item><el-button type="text" icon="el-icon-edit" @click="toUpdate(row.roleId)">修改</el-button></el-dropdown-item>

<el-dropdown-item><el-button type="text" icon="el-icon-delete" @click="toDelete(row.roleId)">删除</el-button></el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

<el-pagination

class="pagination"

:current-page="page.currentPage"

:page-sizes="[10,20,30,50]"

:page-size="page.pageSize"

layout="total, sizes, prev, pager, next, jumper"

:total="dataPage.totalCount"

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

/>

</div>

<!-- 数据表格结束 -->

<!-- 添加弹窗 -->

<el-dialog title="添加角色" width="30%" :visible.sync="addDialog">

<sys-role-add @after="getByPage" />

</el-dialog>

<!-- 添加弹窗结束 -->

<!-- 修改弹窗 -->

<el-dialog title="修改角色" width="30%" :visible.sync="updateDialog">

<sys-role-update :active-id="activeId" @after="getByPage" />

</el-dialog>

<!-- 修改弹窗结束 -->

</div>

</template>

<script>

import sysRoleApi from '@/api/sysRole'

import SysRoleAdd from './sys-role-add'

import SysRoleUpdate from './sys-role-update'

export default {

components: {

SysRoleAdd,

SysRoleUpdate

},

data() {

return {

page: {

currentPage: 1,

pageSize: 10,

params: {}

},

dataPage: {

list: [],

totalCount: 0,

totalPage: 0

},

addDialog: false,

loading: false,

activeId: '',

updateDialog: false,

infoDialog: false

}

},

created() {

this.getByPage()

},

methods: {

handleSizeChange(val) {

this.page.pageSize = val

this.getByPage()

},

toAdd() {

this.addDialog = true

},

toUpdate(id) {

this.activeId = id

this.updateDialog = true

},

toDelete(id) {

this.$confirm('此操作会删除该角色，是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'danger'

}).then(() => {

sysRoleApi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

})

},

changeStatus(row) {

if (row.status === 1) {

this.toEnable(row)

} else {

this.toDisable(row)

}

},

toEnable(row) {

this.$confirm('此操作会启用该角色，是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'success'

}).then(() => {

sysRoleApi.enableById(row.id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

row.status = 0

})

},

toDisable(row) {

this.$confirm('此操作会禁用该角色，是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

sysRoleApi.disableById(row.id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

row.status = 1

})

},

toInfo(id) {

this.activeId = id

this.infoDialog = true

},

handleCurrentChange(val) {

this.page.currentPage = val

this.getByPage()

},

search() {

this.page.currentPage = 1

this.getByPage()

},

closeDialog() {

this.addDialog = false

this.updateDialog = false

this.infoDialog = false

},

getByPage() {

this.closeDialog()

this.loading = true

sysRoleApi.getByPage(this.page).then(res => {

this.dataPage.list = res.data.list

this.dataPage.totalCount = res.data.totalCount

this.dataPage.totalPage = res.data.totalPage

this.loading = false

})

}

}

}

</script>

<style scoped>

.pagination {

margin-top: 15px;

}

.mid-button-group {

margin-bottom: 15px;

}

.button-group {

margin-left: 10px;

}

</style>

#### sys-role-add

vue

<template>

<div>

<el-form ref="form" :model="sysRole" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="角色名">

<el-input v-model="sysRole.roleName" placeholder="请输入角色名" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="排序">

<el-input-number v-model="sysRole.roleSort" style="width: 100%" controls-position="right" :min="1" />

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="sysRole = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysRoleApi from '@/api/sys-role'

export default {

data() {

return {

// 表单对象

sysRole: {}

}

},

methods: {

// 添加角色

add() {

sysRoleApi.save(this.sysRole).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

}

}

}

</script>

<style>

</style>

#### sys-role-update

vue

<template>

<div>

<el-form ref="form" :model="sysRole" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="角色名">

<el-input v-model="sysRole.roleName" placeholder="请输入角色名" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="排序">

<el-input-number v-model="sysRole.roleSort" style="width: 100%" controls-position="right" :min="1" />

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="update">修改</el-button>

<el-button type="warning" @click="sysRole = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysRoleApi from '@/api/sys-role'

export default {

props: {

// 父组件传递的角色ID

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单对象

sysRole: {}

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

methods: {

// 更新角色

update() {

sysRoleApi.update(this.sysRole).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

},

// 根据id查询

getById(id) {

sysRoleApi.get(id).then(res => {

this.sysRole = res.data

})

}

}

}

</script>

<style>

</style>

# 4. 用户角色

用户和角色功能编写完毕后，就可以关联这两张表了。用户和角色 **可以一对一，也可以一对多**，具体看业务场景。本课程采用简单的一对一操作。

尽管这里是一对一，为了保证良好的扩展性，防止后面扩展成一对多场景，因此这里依然建了中间表。

## 4.1 后端

### 4.1.1 SysUserRole

java

package com.jg.pochi.pojo;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* 用户角色关联

\* @Author: 杨德石

\* @Date: 2020/11/13 21:31

\* @Version 1.0

\*/

@Data

public class SysUserRole implements Serializable {

/\*\*

\* 编号

\*/

private Long id;

/\*\*

\* 用户编号

\*/

private Long userId;

/\*\*

\* 角色编号

\*/

private Long roleId;

}

### 4.1.2 SysUserVo

因为我们给用户增加了角色功能，因此这里就需要保存用户所属的角色id。我们新建一个vo类来进行保存

java

package com.jg.pochi.pojo.vo;

import com.jg.pochi.pojo.SysRole;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* 系统用户视图类

\*

\* @Author: 杨德石

\* @Date: 2020/11/13 21:34

\* @Version 1.0

\*/

@Data

public class SysUserVo implements Serializable {

/\*\*

\* id

\*/

private Long id;

/\*\*

\* 用户名

\*/

private String username;

/\*\*

\* 密码

\*/

private String password;

/\*\*

\* 微信的openid

\*/

private String openId;

/\*\*

\* 邮箱

\*/

private String email;

/\*\*

\* 昵称

\*/

private String nickName;

/\*\*

\* 头像

\*/

private String header;

/\*\*

\* 备注

\*/

private String note;

/\*\*

\* 账号启用状态，1是0否

\*/

private Integer status;

/\*\*

\* 创建时间

\*/

private String createTime;

/\*\*

\* 修改时间

\*/

private String updateTime;

/\*\*

\* 最后登录时间

\*/

private String loginTime;

/\*\*

\* 角色

\*/

private SysRole sysRole;

}

VO，即 View Object，视图对象。主要职责是响应给前端显示。一般我们的pojo是对应的表的，但是当涉及到关联查询的时候就不适用了。比如用户要保存角色信息。而像用户密码、删除状态等字段则是不需要展示给用户的，所以应该不返回给前端。这时候就可以使用VO类，定制一个返回实体，只返回需要的数据。

### 4.1.3 SysUserController

修改了实体之后，我们的 getById 方法就需要进行修改了，要返回一个 SysUserVO

java

/\*\*

\* 保存用户

\*

\* @param sysUser

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<?> save(@RequestBody SysUserVo sysUser) {

// 参数校验

if (StringUtils.isBlank(sysUser.getUsername())) {

return new Result<>(ResultEnums.PARAMS\_NULL, "用户名不能为空！");

}

if (StringUtils.isBlank(sysUser.getPassword())) {

return new Result<>(ResultEnums.PARAMS\_NULL, "密码不能为空");

}

sysUserService.save(sysUser);

return new Result<>("添加成功");

}

/\*\*

\* 修改用户

\* 修改接口一般不提供密码修改功能

\*

\* @param sysUser

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<?> update(@RequestBody SysUserVo sysUser) {

sysUserService.update(sysUser);

return new Result<>("修改成功");

}

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysUserVo> get(@PathVariable Long id) {

SysUserVo sysUser = sysUserService.get(id);

return new Result<>(sysUser);

}

同样，在添加和修改时，就需要使用 SysUserVo 来接收了

java

/\*\*

\* 保存

\*

\* @param sysUser

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<Object> save(@RequestBody SysUserVo sysUser) {

String username = sysUser.getUsername();

String password = sysUser.getPassword();

if(StringUtils.isBlank(username)) {

return new Result<>(ResultEnum.PARAMS\_NULL, "用户名不能为空");

}

if(StringUtils.isBlank(password)) {

return new Result<>(ResultEnum.PARAMS\_NULL, "密码不能为空");

}

sysUserService.save(sysUser);

return new Result<>("添加成功！");

}

/\*\*

\* 更新

\*

\* @param sysUser

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<Object> update(@RequestBody SysUserVo sysUser) {

String username = sysUser.getUsername();

if(StringUtils.isBlank(username)) {

return new Result<>(ResultEnum.PARAMS\_NULL, "用户名不能为空");

}

sysUserService.update(sysUser);

return new Result<>("更新成功！");

}

事实上这里还需要创建一个 SysUserDTO，即：Data Transaction Object，数据传输对象。DTO的作用是接收前端的参数，以及在 Controller、Service、Dao 层之间传输。通过查看DTO，就能清晰地看到这个方法到底接收什么参数。

### 4.1.4 SysUserService

java

/\*\*

\* 保存用户

\* @param sysUser

\*/

void save(SysUserVo sysUser);

/\*\*

\* 修改用户

\* @param sysUser

\*/

void update(SysUserVo sysUser);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysUserVo get(Long id);

### 4.1.4 SysUserServiceImpl

同理，实现类也需要改，并且查询的时候得返回带角色的实体

java

@Override

@Transactional(rollbackFor = Exception.class)

public void save(SysUserVo sysUser) {

// 拷贝属性

SysUser user = new SysUser();

BeanUtils.copyProperties(sysUser, user);

// id用雪花算法生成

long userId = idWorker.nextId();

user.setId(userId);

sysUserMapper.save(user);

// 如果角色id存在，存入用户角色表

if (sysUser.getSysRole() != null && sysUser.getSysRole().getRoleId() != null) {

SysUserRole sysUserRole = new SysUserRole();

sysUserRole.setUserId(userId);

sysUserRole.setRoleId(sysUser.getSysRole().getRoleId());

sysUserRoleMapper.save(sysUserRole);

}

}

@Override

public void update(SysUserVo sysUser) {

// 拷贝属性

SysUser user = new SysUser();

BeanUtils.copyProperties(sysUser, user);

sysUserMapper.update(user);

// 不管前端有没有选择角色，我们都把旧的角色信息删掉，如果选择了，就再添加新的角色信息

sysUserRoleMapper.deleteByUserId(user.getId());

// 如果角色id存在，存入用户角色表

if (sysUser.getSysRole() != null && sysUser.getSysRole().getRoleId() != null) {

SysUserRole sysUserRole = new SysUserRole();

sysUserRole.setUserId(user.getId());

sysUserRole.setRoleId(sysUser.getSysRole().getRoleId());

sysUserRoleMapper.save(sysUserRole);

}

}

@Override

public SysUserVo get(Long id) {

SysUser user = sysUserMapper.getById(id);

// 拷贝信息

SysUserVo sysUserVo = new SysUserVo();

BeanUtils.copyProperties(user, sysUserVo);

// 查询角色信息

List<SysUserRole> sysUserRoleList = sysUserRoleMapper.getByUserId(user.getId());

if (!CollectionUtils.isEmpty(sysUserRoleList)) {

// 说明有角色信息，取出角色ID

List<Long> roleIds = sysUserRoleList.stream().map(SysUserRole::getRoleId).collect(Collectors.toList());

// 根据角色ID去查询所有的角色信息

List<SysRole> roleList = sysRoleMapper.getByIds(roleIds);

if (!CollectionUtils.isEmpty(roleList)) {

sysUserVo.setSysRole(roleList.get(0));

}

}

return sysUserVo;

}

### 4.1.6 SysRoleMapper

java

/\*\*

\* 根据ID集合查询所有的角色信息

\* @param roleIds

\* @return

\*/

List<SysRole> getByIds(List<Long> roleIds);

### 4.1.7 SysRoleMapper.xml

xml

<select id="getByIds" resultMap="BaseResultMap">

select role\_id,

role\_name,

role\_sort,

create\_by,

create\_time,

update\_by,

update\_time

from sys\_role

where role\_id in (

<foreach collection="list" item="id" separator=",">

#{id}

</foreach>

)

</select>

### 4.1.8 SysUserRoleMapper

java

package com.jg.pochi.mapper;

import com.jg.pochi.pojo.SysUserRole;

import org.springframework.stereotype.Component;

import java.util.List;

/\*\*

\* 用户角色Mapper

\* @Author: 杨德石

\* @Date: 2020/11/13 21:42

\* @Version 1.0

\*/

@Component

public interface SysUserRoleMapper {

/\*\*

\* 添加

\* @param sysUserRole

\*/

void save(SysUserRole sysUserRole);

/\*\*

\* 根据用户Id删除

\* @param id

\*/

void deleteByUserId(Long id);

/\*\*

\* 根据用户ID查询

\* @param id

\* @return

\*/

List<SysUserRole> getByUserId(Long id);

}

### 4.1.9 SysUserRoleMapper.xml

xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jg.pochi.mapper.SysUserRoleMapper">

<!-- 通用查询映射结果 -->

<resultMap id="BaseResultMap" type="com.jg.pochi.pojo.SysUserRole">

<id column="id" property="id"/>

<result column="user\_id" property="userId"/>

<result column="role\_id" property="roleId"/>

</resultMap>

<insert id="save">

insert into sys\_user\_role(user\_id, role\_id)

VALUES (#{userId}, #{roleId})

</insert>

<delete id="deleteByUserId">

delete

from sys\_user\_role

where user\_id = #{userId}

</delete>

<select id="getByUserId" resultMap="BaseResultMap">

select id, user\_id, role\_id

from sys\_user\_role

where user\_id = #{userId}

</select>

</mapper>

### 4.1.10 SysRoleController

我们在前端需要通过下拉框选择角色，因此需要为角色提供一个查询全部的功能

java

/\*\*

\* 查询所有角色

\* @return

\*/

@RequestMapping(value = "/getAll", method = RequestMethod.GET)

public Result<List<SysRole>> getAll() {

List<SysRole> list = sysRoleService.getAll();

return new Result<>(list);

}

### 4.1.11 SysRoleService

java

/\*\*

\* 查询所有角色

\* @return

\*/

List<SysRole> getAll();

### 4.1.12 SysRoleServiceImpl

java

@Override

public List<SysRole> getAll() {

return sysRoleMapper.getAll();

}

### 4.1.13 SysRoleMapper

java

/\*\*

\* 查询所有角色

\* @return

\*/

List<SysRole> getAll();

### 4.1.14 SysRoleMapper.xml

xml

<select id="getAll" resultMap="BaseResultMap">

select role\_id,

role\_name

from sys\_role

where deleted = 0

order by role\_sort

</select>

## 4.2 前端

### 4.2.1 API

用户的API的我们不需要修改，只需要给角色的API增加一个findAll即可

js

findAll() { // 查询所有

return request({

url: `/${group\_name}/findAll`,

method: 'get'

})

},

### 4.2.2 页面改造

该功能主要是给用户分配权限，因此要在用户添加和修改中操作。

#### sys-user-info.vue

vue

<template>

<div>

<el-form ref="form" :model="sysUser" label-width="80px" size="small">

<el-row>

<el-col :span="24" class="header-container">

<el-image

style="width: 100px; height: 100px"

:src="sysUser.header"

fit="fill"

/>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8">

<el-form-item label="账号">

{{ sysUser.username }}

</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="角色">

{{ sysUser.sysRole.roleName }}

</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="昵称">

{{ sysUser.nickName }}

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="邮箱">

{{ sysUser.email }}

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="启用状态">

{{ sysUser.status === 1 ? '启用': '未启用' }}

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="创建时间">

{{ sysUser.createTime }}

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="更新时间">

{{ sysUser.updateTime }}

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="最后登录">

{{ sysUser.loginTime }}

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="备注">

{{ sysUser.note }}

</el-form-item>

</el-col>

</el-row>

</el-form>

</div>

</template>

<script>

import sysUserApi from '@/api/sys-user'

export default {

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

// 回显用户信息对象

sysUser: {

// 角色信息

sysRole: {}

}

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

methods: {

// 根据id查询

getById(id) {

sysUserApi.get(id).then(res => {

this.sysUser = res.data

})

}

}

}

</script>

<style scoped>

.header-container {

text-align: center;

margin-bottom: 15px;

}

</style>

#### sys-user-add.vue

vue

<template>

<div>

<el-form ref="form" :model="sysUser" :rules="rules" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="账号" prop="username">

<el-input v-model="sysUser.username" placeholder="请输入账号" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="密码" prop="password">

<el-input v-model="sysUser.password" show-password placeholder="请输入密码" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8">

<el-form-item label="昵称" prop="nickName">

<el-input v-model="sysUser.nickName" placeholder="请输入昵称" clearable />

</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="角色">

<el-select v-model="sysUser.sysRole.roleId" style="width: 100%" clearable placeholder="请选择角色">

<el-option

v-for="item in roleList"

:key="item.roleId"

:label="item.roleName"

:value="item.roleId"

/>

</el-select>

</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="邮箱">

<el-input v-model="sysUser.email" placeholder="请输入邮箱" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="备注">

<el-input v-model="sysUser.note" :rows="2" type="textarea" placeholder="请输入备注" clearable />

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="sysUser = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysUserApi from '@/api/sys-user'

import sysRoleApi from '@/api/sys-role'

import md5 from 'js-md5'

export default {

data() {

return {

// 用户表单

sysUser: {

// 角色信息

sysRole: {}

},

// 角色下拉列表

roleList: [],

// 表单校验对象

rules: {

username: [

{ required: true, message: '请输入用户名', trigger: 'blur' },

{ min: 5, message: '用户名最短5位', trigger: 'change' }

],

password: [

{ required: true, message: '请输入密码', trigger: 'blur' },

{ min: 5, message: '密码最短5位', trigger: 'change' }

],

nickName: [

{ required: true, message: '请输入昵称', trigger: 'blur' }

]

}

}

},

created() {

this.getAllRole()

},

methods: {

// 添加

add() {

// 1.密码要md5加密

const form = { ...this.sysUser }

form.password = md5(form.password)

// 2.表单校验

sysUserApi.save(form).then(res => {

this.$message.success(res.msg)

// 关闭对话框，并刷新列表

this.$emit('close')

this.$emit('after')

})

},

// 查询所有角色

getAllRole() {

sysRoleApi.getAll().then(res => {

this.roleList = res.data

})

}

}

}

</script>

<style>

</style>

#### sys-user-update.vue

vue

<template>

<div>

<el-form ref="form" :model="sysUser" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="账号">

<el-input v-model="sysUser.username" readonly placeholder="请输入账号" />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="角色">

<el-select v-model="sysUser.sysRole.roleId" style="width: 100%" clearable placeholder="请选择角色">

<el-option

v-for="item in roleList"

:key="item.roleId"

:label="item.roleName"

:value="item.roleId"

/>

</el-select>

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="昵称">

<el-input v-model="sysUser.nickName" placeholder="请输入昵称" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="邮箱">

<el-input v-model="sysUser.email" placeholder="请输入邮箱" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="备注">

<el-input v-model="sysUser.note" :rows="2" type="textarea" placeholder="请输入备注" clearable />

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="update">修改</el-button>

<el-button type="warning" @click="sysUser = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysUserApi from '@/api/sys-user'

import sysRoleApi from '@/api/sys-role'

export default {

props: {

// 调用方传来的用户ID

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单数据

sysUser: {

// 角色信息

sysRole: {}

},

// 角色下拉列表

roleList: []

}

},

watch: {

activeId: {

// 该属性作用就是让监听器立即生效

immediate: true,

// 处理监听器的实际逻辑

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

created() {

this.getAllRole()

},

methods: {

// 根据id查询用户

getById(id) {

sysUserApi.get(id).then(res => {

this.sysUser = res.data

})

},

// 修改用户

update() {

sysUserApi.update(this.sysUser).then(res => {

this.$message.success(res.msg)

// 关闭对话框，并刷新列表

this.$emit('close')

this.$emit('after')

})

},

// 查询所有角色

getAllRole() {

sysRoleApi.getAll().then(res => {

this.roleList = res.data

})

}

}

}

</script>

<style>

</style>

# 5. 菜单管理

在以往的RBAC中，可能会将菜单表和权限表进行拆分，这样虽然耦合性更低但是操作起来较为不便，尤其是当要调整菜单或者权限结构时，另一张表也需要对应进行调整。

本课程不采用这种方式，而是将权限和菜单存放到一张表中，通过一个状态值标识这条数据是权限还是菜单。

## 5.1 后端

### 5.1.1 SysMenu

java

package com.jg.pochi.pojo;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* 菜单

\*

\* @Author: 杨德石

\* @Date: 2020/11/14 23:55

\* @Version 1.0

\*/

@Data

public class SysMenu implements Serializable {

/\*\*

\* 菜单ID

\*/

private Long menuId;

/\*\*

\* 菜单名

\*/

private String menuName;

/\*\*

\* 父菜单编号

\*/

private Long parentId;

/\*\*

\* 排序值

\*/

private Integer orderNum;

/\*\*

\* 路由地址

\*/

private String routerPath;

/\*\*

\* 组件路径

\*/

protected String componentUrl;

/\*\*

\* 菜单类型，1目录，2菜单，3权限

\*/

private Integer menuType;

/\*\*

\* 是否显示，1是0否

\*/

private Integer visible;

/\*\*

\* 是否启用，1是0否

\*/

private Integer status;

/\*\*

\* 权限标识

\*/

private String permission;

/\*\*

\* 图标

\*/

private String icon;

/\*\*

\* 创建人

\*/

private String createBy;

/\*\*

\* 创建时间

\*/

private String createTime;

/\*\*

\* 修改人

\*/

private String updateBy;

/\*\*

\* 修改时间

\*/

private String updateTime;

/\*\*

\* 是否删除，1是0否

\*/

private Integer deleted;

}

### 5.1.2 SysMenuController

java

package com.jg.pochi.controller;

import com.jg.pochi.common.Page;

import com.jg.pochi.common.Result;

import com.jg.pochi.pojo.SysMenu;

import com.jg.pochi.service.SysMenuService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

/\*\*

\* 菜单控制器

\*

\* @Author: 杨德石

\* @Date: 2020/11/15 0:00

\* @Version 1.0

\*/

@RestController

@RequestMapping("/sysMenu")

public class SysMenuController {

@Autowired

private SysMenuService sysMenuService;

/\*\*

\* 添加菜单

\* @param sysMenu

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<?> save(@RequestBody SysMenu sysMenu) {

sysMenuService.save(sysMenu);

return new Result<>("添加成功");

}

/\*\*

\* 修改菜单

\* @param sysMenu

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<?> update(@RequestBody SysMenu sysMenu) {

sysMenuService.update(sysMenu);

return new Result<>("修改成功");

}

/\*\*

\* 根据id删除

\* @param id

\* @return

\*/

@RequestMapping(value = "/delete/{id}", method = RequestMethod.DELETE)

public Result<?> delete(@PathVariable Long id) {

sysMenuService.delete(id);

return new Result<>("删除成功");

}

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysMenu> get(@PathVariable Long id) {

SysMenu sysMenu = sysMenuService.get(id);

return new Result<>(sysMenu);

}

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

@RequestMapping(value = "/getByPage", method = RequestMethod.POST)

public Result<Page<SysMenu>> getByPage(@RequestBody Page<SysMenu> page) {

page = sysMenuService.getByPage(page);

return new Result<>(page);

}

}

### 5.1.3 SysMenuService

java

package com.jg.pochi.service;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysMenu;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/15 0:03

\* @Version 1.0

\*/

public interface SysMenuService {

/\*\*

\* 添加

\* @param sysMenu

\*/

void save(SysMenu sysMenu);

/\*\*

\* 修改

\* @param sysMenu

\*/

void update(SysMenu sysMenu);

/\*\*

\* 根据id删除

\* @param id

\*/

void delete(Long id);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysMenu get(Long id);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

Page<SysMenu> getByPage(Page<SysMenu> page);

}

### 5.1.4 SysMenuSerivceImpl

java

package com.jg.pochi.service.impl;

import com.jg.pochi.common.Page;

import com.jg.pochi.constant.CoreConstant;

import com.jg.pochi.enums.ResultEnums;

import com.jg.pochi.exception.PochiException;

import com.jg.pochi.mapper.SysMenuMapper;

import com.jg.pochi.pojo.SysMenu;

import com.jg.pochi.pojo.SysUser;

import com.jg.pochi.service.SysMenuService;

import com.jg.pochi.utils.ShiroUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/15 0:03

\* @Version 1.0

\*/

@Service

public class SysMenuServiceImpl implements SysMenuService {

@Autowired

private SysMenuMapper sysMenuMapper;

@Override

public void save(SysMenu sysMenu) {

// 初始化默认值

if (sysMenu.getParentId() == null) {

sysMenu.setParentId(CoreConstant.DEFAULT\_PARENT\_ID);

}

// 根据parentId和name查询该父级菜单下是否存在同名菜单

SysMenu menu = sysMenuMapper.getByParentIdAndName(sysMenu);

// 如果存在，说明菜单已存在

if (menu != null) {

throw new PochiException(ResultEnums.MENU\_EXISTS);

}

// 菜单不存在，入表

SysUser loginUser = ShiroUtils.getLoginUser();

sysMenu.setCreateBy(loginUser.getUsername());

sysMenu.setUpdateBy(loginUser.getUsername());

// 添加

sysMenuMapper.save(sysMenu);

}

@Override

public void update(SysMenu sysMenu) {

// 判断菜单是否存在

// 初始化默认值

if (sysMenu.getParentId() == null) {

sysMenu.setParentId(CoreConstant.DEFAULT\_PARENT\_ID);

}

// 根据parentId和name查询该父级菜单下是否存在同名菜单

SysMenu menu = sysMenuMapper.getByParentIdAndName(sysMenu);

if(menu!=null && !menu.getMenuId().equals(sysMenu.getMenuId())) {

// 如果菜单存在，并且编号不相同，就说明存在了同名的菜单

throw new PochiException(ResultEnums.MENU\_EXISTS);

}

// 设置修改人

SysUser loginUser = ShiroUtils.getLoginUser();

sysMenu.setUpdateBy(loginUser.getUsername());

sysMenuMapper.update(sysMenu);

}

@Override

public void delete(Long id) {

sysMenuMapper.deleteById(id);

}

@Override

public SysMenu get(Long id) {

return sysMenuMapper.getById(id);

}

@Override

public Page<SysMenu> getByPage(Page<SysMenu> page) {

// 设置默认的当前页和每页条数

Integer pageNumber = page.getPageNumber();

if (pageNumber == null || pageNumber < 1) {

pageNumber = 1;

page.setPageNumber(pageNumber);

}

// 查询当前页数据

List<SysMenu> list = sysMenuMapper.getByPage(page);

// 查询总条数

Integer totalCount = sysMenuMapper.countByPage(page);

// 设置值

page.setList(list);

page.setTotalCount(totalCount);

return page;

}

}

### 5.1.5 SysMenuMapper

java

package com.jg.pochi.mapper;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysMenu;

import org.springframework.stereotype.Component;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 13:46

\* @Version 1.0

\*/

@Component

public interface SysMenuMapper {

/\*\*

\* 根据父级菜单ID和名称查询

\* @param sysMenu

\* @return

\*/

SysMenu getByParentIdAndName(SysMenu sysMenu);

/\*\*

\* 添加

\* @param sysMenu

\*/

void save(SysMenu sysMenu);

/\*\*

\* 修改

\* @param sysMenu

\*/

void update(SysMenu sysMenu);

/\*\*

\* 根据id删除

\* @param id

\*/

void deleteById(Long id);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysMenu getById(Long id);

/\*\*

\* 查询总数

\* @param page

\* @return

\*/

Integer countByPage(Page<SysMenu> page);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

List<SysMenu> getByPage(Page<SysMenu> page);

}

### 5.1.6 SysMenuMapper.xml

xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jg.pochi.mapper.SysMenuMapper">

<!-- 通用查询映射结果 -->

<resultMap id="BaseResultMap" type="com.jg.pochi.pojo.SysMenu">

<id column="menu\_id" property="menuId"/>

<result column="menu\_name" property="menuName"/>

<result column="parent\_id" property="parentId"/>

<result column="order\_num" property="orderNum"/>

<result column="router\_path" property="routerPath"/>

<result column="component\_url" property="componentUrl"/>

<result column="menu\_type" property="menuType"/>

<result column="visible" property="visible"/>

<result column="status" property="status"/>

<result column="permission" property="permission"/>

<result column="icon" property="icon"/>

<result column="create\_by" property="createBy"/>

<result column="create\_time" property="createTime"/>

<result column="update\_by" property="updateBy"/>

<result column="update\_time" property="updateTime"/>

<result column="deleted" property="deleted"/>

</resultMap>

<insert id="save">

insert into sys\_menu(menu\_name, parent\_id, order\_num, router\_path, component\_url, menu\_type, visible,

status, permission, icon, create\_by, update\_by)

VALUES (#{menuName}, #{parentId}, #{orderNum}, #{routerPath}, #{componentUrl}, #{menuType}, #{visible},

#{status}, #{permission}, #{icon}, #{createBy}, #{updateBy})

</insert>

<update id="update">

update sys\_menu

<set>

<if test="menuName!=null and menuName!=''">

menu\_name = #{menuName},

</if>

<if test="parentId!=null">

parent\_id = #{parentId},

</if>

<if test="orderNum!=null">

order\_num = #{orderNum},

</if>

<if test="routerPath!=null and routerPath!=''">

router\_path = #{routerPath},

</if>

<if test="componentUrl!=null and componentUrl!=''">

component\_url = #{componentUrl},

</if>

<if test="menuType!=null">

menu\_type = #{menuType},

</if>

<if test="componentUrl!=null and componentUrl!=''">

component\_url = #{componentUrl},

</if>

<if test="visible!=null">

visible = #{visible},

</if>

<if test="status!=null">

status = #{status},

</if>

<if test="permission!=null and permission!=''">

permission = #{permission},

</if>

<if test="icon!=null and icon!=''">

icon = #{icon},

</if>

<if test="updateBy!=null and updateBy!=''">

update\_by = #{updateBy},

</if>

</set>

where menu\_id = #{menuId}

</update>

<update id="deleteById">

update sys\_menu

set deleted = 1

where menu\_id = #{menuId}

</update>

<select id="getByParentIdAndName" resultMap="BaseResultMap">

select menu\_id,

menu\_name

from sys\_menu

where parent\_id = #{parentId}

and menu\_name = #{menuName}

and deleted = 0

</select>

<select id="getById" resultMap="BaseResultMap">

select menu\_id,

menu\_name,

parent\_id,

order\_num,

router\_path,

component\_url,

menu\_type,

visible,

status,

permission,

icon

from sys\_menu

where menu\_id = #{menuId}

</select>

<select id="countByPage" resultType="java.lang.Integer">

select count(\*)

from sys\_menu

where deleted = 0

</select>

<select id="getByPage" resultMap="BaseResultMap">

select menu\_id,

menu\_name,

parent\_id,

order\_num,

router\_path,

component\_url,

menu\_type,

visible,

status,

permission,

icon,

create\_by,

create\_time,

update\_by,

update\_time

from sys\_menu

where deleted = 0

order by order\_num

limit #{index}, #{pageSize}

</select>

</mapper>

## 5.2 前端

### 5.2.1 API定义

在 api 下新建 sysMenu.js

js

import request from '@/utils/request'

var group\_name = 'sysMenu'

export default {

getByPage(page) { // 分页查询

return request({

url: `/${group\_name}/getByPage`,

method: 'post',

data: page

})

},

save(sysMenu) { // 保存

return request({

url: `/${group\_name}/save`,

method: 'post',

data: sysMenu

})

},

get(id) { // 根据id查询

return request({

url: `/${group\_name}/get/${id}`,

method: 'get'

})

},

update(sysMenu) { // 更新

return request({

url: `/${group\_name}/update`,

method: 'put',

data: sysMenu

})

},

findAll() { // 查询所有

return request({

url: `/${group\_name}/findAll`,

method: 'get'

})

},

deleteById(id) { // 根据id删除

return request({

url: `/${group\_name}/delete`,

method: 'put',

data: { menuId: id }

})

}

}

### 5.2.2 路由配置

在 views 下新建目录 system/menu ，目录下分别创建 sys-menu-list.vue、sys-menu-add.vue、sys-menu-update.vue 文件。并在路由文件中配置上菜单的列表页

js

{

path: 'menu',

component: () => import('@/views/system/menu/sys-menu-list'),

name: 'menu',

meta: {

title: '菜单管理',

icon: 'lock'

}

}

### 5.2.3 页面编写

#### sys-menu-list

vue

<template>

<div>

<!-- 添加按钮开始 -->

<div class="button-group">

<el-button type="primary" size="small" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

<!-- 添加按钮结束 -->

<!-- 数据表格开始 -->

<div class="data-table">

<el-table

header-row-class-name="pochi-table-header"

:data="dataPage.list"

stripe

style="width: 100%"

>

<el-table-column prop="menuId" label="菜单编号" width="80px" />

<el-table-column prop="menuName" label="名称" />

<el-table-column prop="orderNum" label="排序" width="60px" />

<el-table-column prop="componentUrl" label="组件路径" />

<el-table-column prop="menuType" label="菜单类型" width="80px">

<template slot-scope="{row}">

<el-tag v-if="row.menuType === 1">目录</el-tag>

<el-tag v-if="row.menuType === 2">菜单</el-tag>

<el-tag v-if="row.menuType === 3">权限</el-tag>

</template>

</el-table-column>

<el-table-column prop="visible" label="显示状态" width="80px">

<template slot-scope="{row}">

<el-tag v-if="row.visible === 1">显示</el-tag>

<el-tag v-if="row.visible === 0" type="info">隐藏</el-tag>

</template>

</el-table-column>

<el-table-column prop="status" label="启用状态" width="80px">

<template slot-scope="{row}">

<el-tag v-if="row.status === 1">启用</el-tag>

<el-tag v-if="row.status === 0" type="info">禁用</el-tag>

</template>

</el-table-column>

<el-table-column prop="permission" label="权限标识" />

<el-table-column prop="icon" label="菜单图标" />

<el-table-column prop="createTime" label="创建时间" />

<el-table-column prop="createBy" label="创建人" width="120px" />

<el-table-column prop="updateTime" label="修改时间" />

<el-table-column prop="updateBy" label="修改人" width="120px" />

<el-table-column label="操作" width="80px">

<template slot-scope="{row}">

<el-dropdown class="handle-button">

<span class="el-dropdown-link">

操作<i class="el-icon-arrow-down el-icon--right" />

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item @click.native="toUpdate(row.menuId)">

<el-button type="text" icon="el-icon-edit">修改</el-button>

</el-dropdown-item>

<el-dropdown-item @click.native="toDelete(row.menuId)">

<el-button type="text" icon="el-icon-delete">删除</el-button>

</el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

</div>

<!-- 数据表格结束 -->

<!-- 添加弹窗 -->

<el-dialog

title="添加菜单"

:visible.sync="addDialog"

width="30%"

>

<sys-menu-add @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 添加弹窗结束 -->

<!-- 修改弹窗 -->

<el-dialog

title="修改菜单"

:visible.sync="updateDialog"

width="30%"

>

<sys-menu-update :active-id="activeId" @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 修改弹窗结束 -->

</div>

</template>

<script>

import sysMenuApi from '@/api/sys-menu'

import sysMenuAdd from './sys-menu-add'

import sysMenuUpdate from './sys-menu-update'

import SysMenuAdd from './sys-menu-add.vue'

export default {

components: {

sysMenuAdd,

sysMenuUpdate

},

data() {

SysMenuAdd

return {

// 数据分页对象

dataPage: {},

// 查询分页对象

page: {

pageNumber: 1,

pageSize: 50,

// 查询参数对象

params: {}

},

// 控制添加弹窗显示

addDialog: false,

// 控制修改弹窗显示

updateDialog: false,

// 当前点击的菜单编号

activeId: ''

}

},

created() {

this.getByPage()

},

methods: {

// 打开添加弹窗

toAdd() {

this.addDialog = true

},

// 打开编辑弹窗

toUpdate(id) {

this.activeId = id

this.updateDialog = true

},

// 删除

toDelete(id) {

this.$confirm('是否删除该菜单?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'error'

}).then(() => {

sysMenuApi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

})

},

// 关闭弹窗

closeDialog() {

this.addDialog = false

this.updateDialog = false

},

// 分页查询

getByPage() {

sysMenuApi.getByPage(this.page).then(res => {

this.dataPage = res.data

})

}

}

}

</script>

<style>

</style>

#### sys-menu-add

页面

vue

<template>

<div>

<el-form :model="sysMenu" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="上级菜单">

<el-input v-model="sysMenu.parentId" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="菜单类型">

<el-radio-group v-model="sysMenu.menuType">

<el-radio :label="1">目录</el-radio>

<el-radio :label="2">菜单</el-radio>

<el-radio :label="3">权限</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

<el-row v-show="sysMenu.menuType != 3" :gutter="20">

<el-col :span="24">

<el-form-item label="菜单图标">

<el-input v-model="sysMenu.icon" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row v-show="sysMenu.menuType === 2" :gutter="20">

<el-col :span="24">

<el-form-item label="组件路径">

<el-input v-model="sysMenu.componentUrl" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="菜单名称">

<el-input v-model="sysMenu.menuName" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="显示排序">

<el-input-number v-model="sysMenu.orderNum" clearable controls-position="right" style="width: 100%" />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col v-show="sysMenu.menuType != 3" :span="12">

<el-form-item label="路由地址">

<el-input v-model="sysMenu.routerPath" clearable />

</el-form-item>

</el-col>

<el-col v-show="sysMenu.menuType != 1" :span="12">

<el-form-item label="权限标识">

<el-input v-model="sysMenu.permission" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col v-show="sysMenu.menuType != 3" :span="12">

<el-form-item label="显示状态">

<el-radio-group v-model="sysMenu.visible">

<el-radio :label="1">显示</el-radio>

<el-radio :label="0">隐藏</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="启用状态">

<el-radio-group v-model="sysMenu.status">

<el-radio :label="1">启用</el-radio>

<el-radio :label="0">禁用</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="sysMenu = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysMenuApi from '@/api/sys-menu'

export default {

data() {

return {

// 表单对象

sysMenu: {

// 菜单类型，1目录，2菜单，3权限

menuType: 1

}

}

},

methods: {

// 添加

add() {

sysMenuApi.save(this.sysMenu).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

}

}

}

</script>

<style>

</style>

#### sys-menu-update

vue

<template>

<div> <el-form :model="sysMenu" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="上级菜单">

<el-input v-model="sysMenu.parentId" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="菜单类型">

<el-radio-group v-model="sysMenu.menuType">

<el-radio :label="1">目录</el-radio>

<el-radio :label="2">菜单</el-radio>

<el-radio :label="3">权限</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

<el-row v-show="sysMenu.menuType != 3" :gutter="20">

<el-col :span="24">

<el-form-item label="菜单图标">

<el-input v-model="sysMenu.icon" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row v-show="sysMenu.menuType === 2" :gutter="20">

<el-col :span="24">

<el-form-item label="组件路径">

<el-input v-model="sysMenu.componentUrl" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="菜单名称">

<el-input v-model="sysMenu.menuName" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="显示排序">

<el-input-number v-model="sysMenu.orderNum" clearable controls-position="right" style="width: 100%" />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col v-show="sysMenu.menuType != 3" :span="12">

<el-form-item label="路由地址">

<el-input v-model="sysMenu.routerPath" clearable />

</el-form-item>

</el-col>

<el-col v-show="sysMenu.menuType != 1" :span="12">

<el-form-item label="权限标识">

<el-input v-model="sysMenu.permission" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col v-show="sysMenu.menuType != 3" :span="12">

<el-form-item label="显示状态">

<el-radio-group v-model="sysMenu.visible">

<el-radio :label="1">显示</el-radio>

<el-radio :label="0">隐藏</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="启用状态">

<el-radio-group v-model="sysMenu.status">

<el-radio :label="1">启用</el-radio>

<el-radio :label="0">禁用</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="update">修改</el-button>

<el-button type="warning" @click="sysMenu = {}">重置</el-button>

</el-form-item>

</el-form></div>

</template>

<script>

import sysMenuApi from '@/api/sys-menu'

export default {

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单对象

sysMenu: {}

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

methods: {

// 根据id查询

getById(id) {

sysMenuApi.get(id).then(res => {

this.sysMenu = res.data

})

},

// 修改

update() {

sysMenuApi.update(this.sysMenu).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

}

}

}

</script>

<style>

</style>

## 5.3 树形结构

我们上面的功能中，尽管已经有了基本的增删改查，但是还存在两个问题

* 列表页没有树形结构
* 添加和修改功能中选择上级菜单没有树形结构

### 5.3.1 树形表格

查看Elementui文档，有树形表格组件，我们按照组件的要求提供接口即可。

不要提供hasChildren字段。否则可能无法正常渲染树结构，这里是个坑

#### SysMenuVo

java

package com.jg.pochi.pojo.vo;

import lombok.Data;

import java.io.Serializable;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/15 22:57

\* @Version 1.0

\*/

@Data

public class SysMenuVo implements Serializable {

/\*\*

\* 菜单ID

\*/

private Long menuId;

/\*\*

\* 菜单名

\*/

private String menuName;

/\*\*

\* 父菜单编号

\*/

private Long parentId;

/\*\*

\* 排序值

\*/

private Integer orderNum;

/\*\*

\* 路由地址

\*/

private String routerPath;

/\*\*

\* 组件路径

\*/

protected String componentUrl;

/\*\*

\* 菜单类型，1目录，2菜单，3权限

\*/

private Integer menuType;

/\*\*

\* 是否显示，1是0否

\*/

private Integer visible;

/\*\*

\* 是否启用，1是0否

\*/

private Integer status;

/\*\*

\* 权限标识

\*/

private String permission;

/\*\*

\* 图标

\*/

private String icon;

/\*\*

\* 创建人

\*/

private String createBy;

/\*\*

\* 创建时间

\*/

private String createTime;

/\*\*

\* 修改人

\*/

private String updateBy;

/\*\*

\* 修改时间

\*/

private String updateTime;

/\*\*

\* 子节点

\*/

private List<SysMenuVo> children;

}

#### SysMenuController

java

/\*\*

\* 树形查询

\* @return

\*/

@RequestMapping(value = "/getTreeList", method = RequestMethod.GET)

public Result<List<SysMenuVo>> getTreeList() {

List<SysMenuVo> list = sysMenuService.getTreeList();

return new Result<>(list);

}

#### SysMenuService

java

/\*\*

\* 树形查询

\* @return

\*/

List<SysMenuVo> getTreeList();

#### SysMenuServiceImpl

java

@Override

public List<SysMenuVo> getTreeList() {

// 查询出所有的菜单

List<SysMenu> menuList = sysMenuMapper.getAll();

// 过滤出所有顶级菜单

return menuList.stream()

// 只要父级菜单是0的就是顶级菜单

.filter(e -> e.getParentId().equals(CoreConstant.DEFAULT\_PARENT\_ID))

// 将顶级菜单转换成我们的视图类

.map(e -> {

SysMenuVo sysMenuVo = new SysMenuVo();

BeanUtils.copyProperties(e, sysMenuVo);

return sysMenuVo;

})

// 根据顶级菜单的ID，递归从剩余的列表中找子菜单

.map(e -> {

e.setChildren(getChildren(e, menuList));

// 处理完之后，判断子菜单是否为空，如果为空，给一个null

if (CollectionUtils.isEmpty(e.getChildren())) {

e.setChildren(null);

}

return e;

}).collect(Collectors.toList());

}

/\*\*

\* 递归构造树形菜单

\*

\* @param sysMenu

\* @param menuList

\* @return

\*/

private List<SysMenuVo> getChildren(SysMenuVo sysMenu, List<SysMenu> menuList) {

// 第一步，直接找到sysMenu的子菜单

List<SysMenuVo> childrenList = menuList.stream().filter(e -> e.getParentId().equals(sysMenu.getMenuId()))

// 第二步，把子菜单每一项转成SysMenuVo

.map(e -> {

SysMenuVo sysMenuVo = new SysMenuVo();

BeanUtils.copyProperties(e, sysMenuVo);

return sysMenuVo;

})

// 第三步，递归找到本次获取到的所有子菜单的子菜单

.map(e -> {

e.setChildren(getChildren(e, menuList));

if (CollectionUtils.isEmpty(e.getChildren())) {

e.setChildren(null);

}

return e;

}).collect(Collectors.toList());

// 判断childrenList是不是空，如果是就返回null

if (CollectionUtils.isEmpty(childrenList)) {

// 这一步其实是过滤掉空集合

return null;

}

return childrenList;

}

#### API

api中我们新增一条树形查询接口

js

/\*\*

\* 树形查询菜单

\*/

getTreeList() {

return request({

url: `/${groupName}/getTreeList`,

methods: 'get'

})

}

#### 页面

vue

<template>

<div>

<!-- 添加按钮开始 -->

<div class="button-group">

<el-button type="primary" size="small" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

<!-- 添加按钮结束 -->

<!-- 数据表格开始 -->

<div class="data-table">

<el-table

header-row-class-name="pochi-table-header"

:data="treeList"

stripe

row-key="menuId"

style="width: 100%"

>

<el-table-column prop="menuId" label="菜单编号" width="100px" />

<el-table-column prop="menuName" label="名称" />

<el-table-column prop="orderNum" label="排序" width="60px" />

<el-table-column prop="componentUrl" label="组件路径" />

<el-table-column prop="menuType" label="菜单类型" width="80px">

<template slot-scope="{row}">

<el-tag v-if="row.menuType === 1">目录</el-tag>

<el-tag v-if="row.menuType === 2">菜单</el-tag>

<el-tag v-if="row.menuType === 3">权限</el-tag>

</template>

</el-table-column>

<el-table-column prop="visible" label="显示状态" width="80px">

<template slot-scope="{row}">

<el-tag v-if="row.visible === 1">显示</el-tag>

<el-tag v-if="row.visible === 0" type="info">隐藏</el-tag>

</template>

</el-table-column>

<el-table-column prop="status" label="启用状态" width="80px">

<template slot-scope="{row}">

<el-tag v-if="row.status === 1">启用</el-tag>

<el-tag v-if="row.status === 0" type="info">禁用</el-tag>

</template>

</el-table-column>

<el-table-column prop="permission" label="权限标识" />

<el-table-column prop="icon" label="菜单图标" />

<el-table-column prop="createTime" label="创建时间" />

<el-table-column prop="createBy" label="创建人" width="120px" />

<el-table-column prop="updateTime" label="修改时间" />

<el-table-column prop="updateBy" label="修改人" width="120px" />

<el-table-column label="操作" width="80px">

<template slot-scope="{row}">

<el-dropdown class="handle-button">

<span class="el-dropdown-link">

操作<i class="el-icon-arrow-down el-icon--right" />

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item @click.native="toUpdate(row.menuId)">

<el-button type="text" icon="el-icon-edit">修改</el-button>

</el-dropdown-item>

<el-dropdown-item @click.native="toDelete(row.menuId)">

<el-button type="text" icon="el-icon-delete">删除</el-button>

</el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

</div>

<!-- 数据表格结束 -->

<!-- 添加弹窗 -->

<el-dialog

title="添加菜单"

:visible.sync="addDialog"

width="30%"

>

<sys-menu-add v-if="addDialog" @after="getTreeList" @close="closeDialog" />

</el-dialog>

<!-- 添加弹窗结束 -->

<!-- 修改弹窗 -->

<el-dialog

title="修改菜单"

:visible.sync="updateDialog"

width="30%"

>

<sys-menu-update v-if="updateDialog" :active-id="activeId" @after="getTreeList" @close="closeDialog" />

</el-dialog>

<!-- 修改弹窗结束 -->

</div>

</template>

<script>

import sysMenuApi from '@/api/sys-menu'

import sysMenuAdd from './sys-menu-add'

import sysMenuUpdate from './sys-menu-update'

import SysMenuAdd from './sys-menu-add.vue'

export default {

components: {

sysMenuAdd,

sysMenuUpdate

},

data() {

SysMenuAdd

return {

// 树形结构

treeList: [],

// 控制添加弹窗显示

addDialog: false,

// 控制修改弹窗显示

updateDialog: false,

// 当前点击的菜单编号

activeId: ''

}

},

created() {

this.getTreeList()

},

methods: {

// 打开添加弹窗

toAdd() {

this.addDialog = true

},

// 打开编辑弹窗

toUpdate(id) {

this.activeId = id

this.updateDialog = true

},

// 删除

toDelete(id) {

this.$confirm('是否删除该菜单?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'error'

}).then(() => {

sysMenuApi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getTreeList()

})

})

},

// 关闭弹窗

closeDialog() {

this.addDialog = false

this.updateDialog = false

},

// 树状查询

getTreeList() {

sysMenuApi.getTreeList().then(res => {

this.treeList = res.data

})

}

}

}

</script>

<style>

</style>

### 5.3.2 下拉选择树

添加功能需要有下拉选择树。翻遍了ElementUI文档，我们也没找到下拉选择树，因此我们需要使用第三方的下拉树，这里推荐 @riophae/vue-treeselect

安装

sh

npm i @riophae/vue-treeselect

安装完毕后引入即可使用。

js

import Treeselect from '@riophae/vue-treeselect'

import '@riophae/vue-treeselect/dist/vue-treeselect.css'

components: { Treeselect },

页面中使用

vue

<template>

<treeselect

v-model="menu.parentId"

:options="menuOptions"

:normalizer="normalizer"

:show-count="true"

placeholder="选择上级菜单"

/>

</template>

<script>

import Treeselect from '@riophae/vue-treeselect'

import '@riophae/vue-treeselect/dist/vue-treeselect.css'

export default {

name: 'Menu',

components: { Treeselect },

data() {

return {

// 菜单

menu: {},

// 菜单表格树数据

menuList: [],

// 菜单树选项

menuOptions: []

}

},

methods: {

/\*\* 转换菜单数据结构 \*/

normalizer(node) {

if (node.children && !node.children.length) {

delete node.children

}

return {

id: node.menuId,

label: node.menuName,

children: node.children

}

},

/\*\* 查询菜单下拉树结构 \*/

getTreeselect() {

listMenu().then(response => {

this.menuOptions = []

const menu = { menuId: 0, menuName: '主类目', children: [] }

menu.children = handleTree(response.data, 'menuId')

this.menuOptions.push(menu)

})

}

}

}

</script>

</script>

#### 添加页

vue

<template>

<div>

<el-form :model="sysMenu" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="上级菜单">

<treeselect

v-model="sysMenu.parentId"

:options="menuOptions"

:normalizer="normalizer"

:show-count="true"

placeholder="请选择上级菜单"

/>

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="菜单类型">

<el-radio-group v-model="sysMenu.menuType">

<el-radio :label="1">目录</el-radio>

<el-radio :label="2">菜单</el-radio>

<el-radio :label="3">权限</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

<el-row v-show="sysMenu.menuType != 3" :gutter="20">

<el-col :span="24">

<el-form-item label="菜单图标">

<el-input v-model="sysMenu.icon" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row v-show="sysMenu.menuType === 2" :gutter="20">

<el-col :span="24">

<el-form-item label="组件路径">

<el-input v-model="sysMenu.componentUrl" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="菜单名称">

<el-input v-model="sysMenu.menuName" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="显示排序">

<el-input-number v-model="sysMenu.orderNum" clearable controls-position="right" style="width: 100%" />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col v-show="sysMenu.menuType != 3" :span="12">

<el-form-item label="路由地址">

<el-input v-model="sysMenu.routerPath" clearable />

</el-form-item>

</el-col>

<el-col v-show="sysMenu.menuType != 1" :span="12">

<el-form-item label="权限标识">

<el-input v-model="sysMenu.permission" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col v-show="sysMenu.menuType != 3" :span="12">

<el-form-item label="显示状态">

<el-radio-group v-model="sysMenu.visible">

<el-radio :label="1">显示</el-radio>

<el-radio :label="0">隐藏</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="启用状态">

<el-radio-group v-model="sysMenu.status">

<el-radio :label="1">启用</el-radio>

<el-radio :label="0">禁用</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="sysMenu = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysMenuApi from '@/api/sys-menu'

import Treeselect from '@riophae/vue-treeselect'

import '@riophae/vue-treeselect/dist/vue-treeselect.css'

export default {

components: {

Treeselect

},

data() {

return {

// 表单对象

sysMenu: {

// 菜单类型，1目录，2菜单，3权限

menuType: 1

},

// 菜单树结构

menuOptions: []

}

},

created() {

this.getTreeSelect()

},

methods: {

// 添加

add() {

sysMenuApi.save(this.sysMenu).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

},

// 设置属性下拉框结构

normalizer(node) {

if (!node.children) {

delete node.children

}

return {

id: node.menuId,

label: node.menuName,

children: node.children

}

},

// 加载树形数据

getTreeSelect() {

sysMenuApi.getTreeList().then(res => {

// 给menuoptions清空

this.menuOptions = []

// 设置顶级菜单的父级选项

const menu = { menuId: 0, menuName: '根菜单', children: [] }

menu.children = res.data

this.menuOptions.push(menu)

})

}

}

}

</script>

<style>

.vue-treeselect\_\_control {

border: 1px solid #dcdfe6 !important;

}

</style>

#### 修改页

vue

<template>

<div> <el-form :model="sysMenu" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="上级菜单">

<treeselect

v-model="sysMenu.parentId"

:options="menuOptions"

:normalizer="normalizer"

:show-count="true"

placeholder="请选择上级菜单"

/>

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="菜单类型">

<el-radio-group v-model="sysMenu.menuType">

<el-radio :label="1">目录</el-radio>

<el-radio :label="2">菜单</el-radio>

<el-radio :label="3">权限</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

<el-row v-show="sysMenu.menuType != 3" :gutter="20">

<el-col :span="24">

<el-form-item label="菜单图标">

<el-input v-model="sysMenu.icon" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row v-show="sysMenu.menuType === 2" :gutter="20">

<el-col :span="24">

<el-form-item label="组件路径">

<el-input v-model="sysMenu.componentUrl" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="菜单名称">

<el-input v-model="sysMenu.menuName" clearable />

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="显示排序">

<el-input-number v-model="sysMenu.orderNum" clearable controls-position="right" style="width: 100%" />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col v-show="sysMenu.menuType != 3" :span="12">

<el-form-item label="路由地址">

<el-input v-model="sysMenu.routerPath" clearable />

</el-form-item>

</el-col>

<el-col v-show="sysMenu.menuType != 1" :span="12">

<el-form-item label="权限标识">

<el-input v-model="sysMenu.permission" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col v-show="sysMenu.menuType != 3" :span="12">

<el-form-item label="显示状态">

<el-radio-group v-model="sysMenu.visible">

<el-radio :label="1">显示</el-radio>

<el-radio :label="0">隐藏</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="启用状态">

<el-radio-group v-model="sysMenu.status">

<el-radio :label="1">启用</el-radio>

<el-radio :label="0">禁用</el-radio>

</el-radio-group>

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="update">修改</el-button>

<el-button type="warning" @click="sysMenu = {}">重置</el-button>

</el-form-item>

</el-form></div>

</template>

<script>

import sysMenuApi from '@/api/sys-menu'

import Treeselect from '@riophae/vue-treeselect'

import '@riophae/vue-treeselect/dist/vue-treeselect.css'

export default {

components: {

Treeselect

},

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单对象

sysMenu: {},

// 菜单树结构

menuOptions: []

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

created() {

this.getTreeSelect()

},

methods: {

// 根据id查询

getById(id) {

sysMenuApi.get(id).then(res => {

this.sysMenu = res.data

})

},

// 修改

update() {

sysMenuApi.update(this.sysMenu).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

},

// 设置属性下拉框结构

normalizer(node) {

if (!node.children) {

delete node.children

}

return {

id: node.menuId,

label: node.menuName,

children: node.children

}

},

// 加载树形数据

getTreeSelect() {

sysMenuApi.getTreeList().then(res => {

// 给menuoptions清空

this.menuOptions = []

// 设置顶级菜单的父级选项

const menu = { menuId: 0, menuName: '根菜单', children: [] }

menu.children = res.data

this.menuOptions.push(menu)

})

}

}

}

</script>

<style>

.vue-treeselect\_\_control {

border: 1px solid #dcdfe6 !important;

}

</style>

## 5.4 图标选择器

我们的菜单功能目前还缺少图标。考虑到用户体验，图标得让用户选择，因此我们这里需要手写一个图标选择器。

### 5.4.1 组件封装

components 目录下新建菜单 SelectIcon ，并在下面创建一个 index.vue。将 views/icons 里所有的代码拷贝过去进行修改

vue

<template>

<div class="icons-container">

<el-tabs type="border-card">

<el-tab-pane label="Icons">

<div class="grid">

<div v-for="item of svgIcons" :key="item" @click="handleClipboard(generateIconCode(item),$event)">

<el-tooltip placement="top">

<div slot="content">

{{ generateIconCode(item) }}

</div>

<div class="icon-item">

<svg-icon :icon-class="item" class-name="disabled" />

<span>{{ item }}</span>

</div>

</el-tooltip>

</div>

</div>

</el-tab-pane>

<el-tab-pane label="Element-UI Icons">

<div class="grid">

<div v-for="item of elementIcons" :key="item" @click="handleClipboard(generateElementIconCode(item),$event)">

<el-tooltip placement="top">

<div slot="content">

{{ generateElementIconCode(item) }}

</div>

<div class="icon-item">

<i :class="'el-icon-' + item" />

<span>{{ item }}</span>

</div>

</el-tooltip>

</div>

</div>

</el-tab-pane>

</el-tabs>

</div>

</template>

<script>

import clipboard from '@/utils/clipboard'

import svgIcons from './svg-icons'

import elementIcons from './element-icons'

export default {

name: 'Icons',

data() {

return {

svgIcons,

elementIcons

}

},

methods: {

generateIconCode(symbol) {

return `<svg-icon icon-class="${symbol}" />`

},

generateElementIconCode(symbol) {

return `<i class="el-icon-${symbol}" />`

},

handleClipboard(text, event) {

clipboard(text, event)

}

}

}

</script>

<style lang="scss" scoped>

.icons-container {

margin: 10px 20px 0;

overflow: hidden;

.grid {

position: relative;

display: grid;

grid-template-columns: repeat(auto-fill, minmax(120px, 1fr));

}

.icon-item {

margin: 20px;

height: 85px;

text-align: center;

width: 100px;

float: left;

font-size: 30px;

color: #24292e;

cursor: pointer;

}

span {

display: block;

font-size: 16px;

margin-top: 10px;

}

.disabled {

pointer-events: none;

}

}

</style>

大致封装完毕之后，开始引入测试

### 5.4.2 组件测试

在添加菜单的位置进行组件测试

html

<el-popover

placement="bottom"

title="选择图标"

width="400"

trigger="click"

>

<div class="icon-content">

<select-icon />

</div>

<el-button slot="reference">click 激活</el-button>

</el-popover>

css

<style scoped>

.icon-content {

max-height: 400px;

overflow-y: scroll;

}

</style>

测试效果基本没问题，下面就是对组件样式进行微调了

### 5.4.3 样式调整

vue-element-admin 提供了两种icon方案，因此我们需要同时兼容这两种

vue

<template>

<div class="icons-container">

<div class="grid">

<div v-for="item of elementIcons" :key="item">

<div class="icon-item" @click="selectIcon('el-icon-' + item)">

<i :class="'el-icon-' + item" />

</div>

</div>

</div>

<div class="grid">

<div v-for="item of svgIcons" :key="item">

<div class="icon-item" @click="selectIcon(item)">

<svg-icon :icon-class="item" class-name="disabled" />

</div>

</div>

</div>

</div>

</template>

<script>

import svgIcons from './svg-icons'

import elementIcons from './element-icons'

export default {

name: 'Icons',

data() {

return {

svgIcons,

elementIcons

}

},

methods: {

selectIcon(icon) {

this.$emit('checkIcon', icon)

}

}

}

</script>

<style lang="scss" scoped>

.icons-container {

overflow: hidden;

.grid {

position: relative;

display: grid;

grid-template-columns: repeat(auto-fill, minmax(40px, 1fr));

}

.icon-item {

height: 40px;

text-align: center;

width: 40px;

float: left;

font-size: 30px;

color: #24292e;

cursor: pointer;

}

span {

display: block;

font-size: 16px;

margin-top: 10px;

}

.disabled {

pointer-events: none;

}

}

</style>

### 5.4.4 组件调用

调用方代码

vue

<el-popover

placement="bottom"

title="选择图标"

width="400"

trigger="click"

>

<div class="icon-content">

<select-icon @checkIcon="checkIcon" />

</div>

<el-button v-if="!menu.menuIcon" slot="reference">选择</el-button>

<el-button v-else-if="menu.menuIcon.indexOf('el-icon') >= 0" slot="reference" :icon="menu.menuIcon">选择</el-button>

<el-button v-else slot="reference"><svg-icon :icon-class="menu.menuIcon" class-name="disabled" /> 选择</el-button>

</el-popover>

checkIcon(icon) {

console.log(icon)

this.$set(this.menu, 'menuIcon', icon)

},

### 5.4.5 最终封装

至此我们封装基本告一段落。我们不难发现，事实上我们需要的仅仅是 checkIcon 这个事件而已，至于你页面怎么交互，这是组件的工作，因此我们的业务代码也不必要关心按钮的切换。

我们可以把按钮也封装到组件中

组件代码

vue

<template>

<div>

<el-popover

v-model="visible"

placement="bottom"

title="选择图标"

width="400"

trigger="manual"

>

<div class="icon-content">

<div class="icons-container">

<div class="grid">

<div v-for="item of elementIcons" :key="item">

<div class="icon-item" @click="selectIcon('el-icon-' + item)">

<i :class="'el-icon-' + item" />

</div>

</div>

</div>

<div class="grid">

<div v-for="item of svgIcons" :key="item">

<div class="icon-item" @click="selectIcon(item)">

<svg-icon :icon-class="item" class-name="disabled" />

</div>

</div>

</div>

</div>

</div>

<el-button v-if="!icon" slot="reference" @click="visible = !visible">选择</el-button>

<el-button v-else-if="icon.indexOf('el-icon') >= 0" slot="reference" :icon="icon" @click="visible = !visible">选择</el-button>

<el-button v-else slot="reference" @click="visible = !visible"><svg-icon :icon-class="icon" class-name="disabled" /> 选择</el-button>

</el-popover>

</div>

</template>

<script>

import svgIcons from './svg-icons'

import elementIcons from './element-icons'

export default {

name: 'Icons',

data() {

return {

svgIcons,

elementIcons,

icon: null,

visible: false

}

},

methods: {

selectIcon(icon) {

this.visible = !this.visible

this.icon = icon

this.$emit('checkIcon', icon)

}

}

}

</script>

<style lang="scss" scoped>

.icons-container {

overflow: hidden;

.grid {

position: relative;

display: grid;

grid-template-columns: repeat(auto-fill, minmax(40px, 1fr));

}

.icon-item {

height: 40px;

text-align: center;

width: 40px;

float: left;

font-size: 30px;

color: #24292e;

cursor: pointer;

}

span {

display: block;

font-size: 16px;

margin-top: 10px;

}

.disabled {

pointer-events: none;

}

}

.icon-content {

max-height: 400px;

overflow-y: scroll;

}

</style>

调用方

html

<select-icon @checkIcon="checkIcon" />

### 5.5.6 菜单页面

html

<el-table-column

prop="icon"

label="菜单图标"

>

<template slot-scope="{row}">

<i v-if="row.icon.indexOf('el-icon') >= 0" :class="row.icon" />

<svg-icon v-else :icon-class="row.icon" class-name="disabled" />

</template>

</el-table-column>

# 6. 角色权限

菜单表功能我们编写完毕后，但只是简单的列表CRUD功能，因此接下来要编写角色权限功能，将角色和菜单绑定起来，*根据角色分配菜单*。

通常情况下，是角色和权限多对多对应，**某些场景也会存在用户和权限对应的情况**，我们不考虑后者，只做角色权限功能。

## 6.1 后端

### 6.1.1 SysRoleMenu

java

package com.jg.pochi.pojo;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/16 21:04

\* @Version 1.0

\*/

@Data

public class SysRoleMenu implements Serializable {

/\*\*

\* 主键，自增

\*/

private Long id;

/\*\*

\* 角色编号

\*/

private Long roleId;

/\*\*

\* 菜单编号

\*/

private Long menuId;

}

### 6.1.2 SysRoleVo

和用户角色相同，我们的角色实体也需要加上权限属性。因为是多对多关系，所以角色中的权限属性需要使用集合来表示。而权限信息往往是不需要在详情页展示的，因此我们这里只**存权限的编号**。



java

package com.jg.pochi.pojo.vo;

import lombok.Data;

import java.io.Serializable;

import java.util.List;

/\*\*

\* 角色视图类

\*

\* @Author: 杨德石

\* @Date: 2020/11/16 21:07

\* @Version 1.0

\*/

@Data

public class SysRoleVo implements Serializable {

/\*\*

\* 角色ID，自增

\*/

private Long roleId;

/\*\*

\* 角色名

\*/

private String roleName;

/\*\*

\* 排序值，越小越靠前

\*/

private Integer roleSort;

/\*\*

\* 菜单ID集合

\*/

private List<Long> authIds;

}

### 6.1.3 SysRoleMenuMapper

接下来我们开始编写权限功能。

为角色授予权限的方式跟为用户赋予角色的做法类似，都是在添加和修改功能上进行的。

添加角色时，直接把权限ID存到数据库中。

修改角色时，先把角色旧的数据删除，再添加新的角色。

因此 SysRoleMenuMapper 中需要加上 根据角色id查询 以及 根据角色id删除 的接口

java

package com.jg.pochi.mapper;

import com.jg.pochi.pojo.SysRoleMenu;

import org.springframework.stereotype.Component;

import java.util.List;

/\*\*

\* 角色权限Mapper

\*

\* @Author: 杨德石

\* @Date: 2020/11/13 21:42

\* @Version 1.0

\*/

@Component

public interface SysRoleMenuMapper {

/\*\*

\* 批量插入

\*

\* @param roleMenuList

\*/

void saveBatch(List<SysRoleMenu> roleMenuList);

/\*\*

\* 根据角色ID删除

\* @param roleId

\*/

void deleteByRoleId(Long roleId);

/\*\*

\* 根据角色ID查询

\* @param id

\* @return

\*/

List<SysRoleMenu> getByRoleId(Long id);

}

### 6.1.4 SysRoleMenuMapper.xml

xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jg.pochi.mapper.SysRoleMenuMapper">

<!-- 通用查询映射结果 -->

<resultMap id="BaseResultMap" type="com.jg.pochi.pojo.SysRoleMenu">

<id column="id" property="id"/>

<result column="menu\_id" property="menuId"/>

<result column="role\_id" property="roleId"/>

</resultMap>

<insert id="saveBatch">

insert into sys\_role\_menu(role\_id, menu\_id) VALUES

<foreach collection="list" item="roleMenu" separator=",">

(#{roleMenu.roleId}, #{roleMenu.menuId})

</foreach>

</insert>

<delete id="deleteByRoleId">

delete

from sys\_role\_menu

where role\_id = #{roleId}

</delete>

<select id="getByRoleId" resultMap="BaseResultMap">

select id, role\_id, menu\_id

from sys\_role\_menu

where role\_id = #{roleId}

</select>

</mapper>

### 6.1.5 SysRoleController

下面开始改造添加和修改接口，同样我们这里就不创建DTO了，直接使用VO接收参数。

java

/\*\*

\* 添加

\* @param sysRole

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<?> save(@RequestBody SysRoleVo sysRole) {

sysRoleService.save(sysRole);

return new Result<>("添加成功");

}

/\*\*

\* 修改

\* @param sysRole

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<?> update(@RequestBody SysRoleVo sysRole) {

sysRoleService.update(sysRole);

return new Result<>("修改成功");

}

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysRoleVo> get(@PathVariable Long id) {

SysRoleVo sysRole = sysRoleService.get(id);

return new Result<>(sysRole);

}

### 6.1.6 SysRoleService

java

/\*\*

\* 添加

\* @param sysRole

\*/

void save(SysRoleVo sysRole);

/\*\*

\* 修改

\* @param sysRole

\*/

void update(SysRoleVo sysRole);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysRoleVo get(Long id);

### 6.1.7 SysRoleServiceImpl

java

@Override

@Transactional(rollbackFor = Exception.class)

public void save(SysRoleVo sysRole) {

// 设置创建人和修改人为用户名

SysUser loginUser = ShiroUtils.getLoginUser();

String username = loginUser.getUsername();

// 创建SysRole对象

SysRole role = new SysRole();

// 拷贝属性

BeanUtils.copyProperties(sysRole, role);

role.setCreateBy(username);

role.setUpdateBy(username);

sysRoleMapper.save(role);

// 下面开始添加角色权限数据

saveRoleMenu(sysRole, role);

}

@Override

@Transactional(rollbackFor = Exception.class)

public void update(SysRoleVo sysRole) {

// 设置更新人

SysUser loginUser = ShiroUtils.getLoginUser();

String username = loginUser.getUsername();

// 创建SysRole对象

SysRole role = new SysRole();

// 拷贝属性

BeanUtils.copyProperties(sysRole, role);

role.setUpdateBy(username);

sysRoleMapper.update(role);

// 删除当前角色的所有权限

sysRoleMenuMapper.deleteByRoleId(role.getRoleId());

// 下面开始添加角色权限数据

saveRoleMenu(sysRole, role);

}

/\*\*

\* 保存角色菜单数据

\* @param sysRole

\* @param role

\*/

private void saveRoleMenu(SysRoleVo sysRole, SysRole role) {

if(!CollectionUtils.isEmpty(sysRole.getAuthIds())) {

// 根据菜单ID集合去构建SysRoleMenu

List<SysRoleMenu> roleMenuList = sysRole.getAuthIds().stream().map(id -> {

SysRoleMenu sysRoleMenu = new SysRoleMenu();

// 设置菜单ID和角色ID

sysRoleMenu.setMenuId(id);

sysRoleMenu.setRoleId(role.getRoleId());

return sysRoleMenu;

}).collect(Collectors.toList());

// 存库

sysRoleMenuMapper.saveBatch(roleMenuList);

}

}

@Override

public SysRoleVo get(Long id) {

SysRole sysRole = sysRoleMapper.get(id);

// 拷贝属性

SysRoleVo vo = new SysRoleVo();

BeanUtils.copyProperties(sysRole, vo);

// 查询这个角色存在的所有权限

List<SysRoleMenu> roleMenuList = sysRoleMenuMapper.getByRoleId(id);

// 如果角色权限集合不为空，取出菜单ID集合

if(!CollectionUtils.isEmpty(roleMenuList)) {

// 取出权限ID集合

List<Long> authIds = roleMenuList.stream().map(SysRoleMenu::getMenuId).collect(Collectors.toList());

vo.setAuthIds(authIds);

}

return vo;

}

## 6.2 前端

### 6.2.1 页面修改

#### sys-role-add

vue

<template>

<div>

<el-form ref="form" :model="sysRole" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="角色名">

<el-input v-model="sysRole.roleName" placeholder="请输入角色名" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="排序">

<el-input-number v-model="sysRole.roleSort" style="width: 100%" controls-position="right" :min="1" />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="权限">

<el-tree

ref="authTree"

:props="{label: 'menuName'}"

:data="menuTree"

show-checkbox

:render-after-expand="false"

node-key="menuId"

/>

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="sysRole = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysRoleApi from '@/api/sys-role'

import sysMenuApi from '@/api/sys-menu'

export default {

data() {

return {

// 表单对象

sysRole: {},

// 菜单树

menuTree: []

}

},

created() {

this.getTreeList()

},

methods: {

// 添加角色

add() {

const checkKeys = this.$refs.authTree.getCheckedKeys()

const halfCheckKeys = this.$refs.authTree.getHalfCheckedKeys()

// 合并选中的数组和半选中的数组

checkKeys.push(...halfCheckKeys)

this.sysRole.authIds = checkKeys

sysRoleApi.save(this.sysRole).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

},

// 树状查询

getTreeList() {

sysMenuApi.getTreeList().then(res => {

this.menuTree = res.data

})

}

}

}

</script>

<style>

</style>

#### sys-role-update

vue

<template>

<div>

<el-form ref="form" :model="sysRole" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="角色名">

<el-input v-model="sysRole.roleName" placeholder="请输入角色名" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="排序">

<el-input-number v-model="sysRole.roleSort" style="width: 100%" controls-position="right" :min="1" />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="权限">

<el-tree

ref="authTree"

:props="{label: 'menuName'}"

:data="menuTree"

show-checkbox

:render-after-expand="false"

node-key="menuId"

/>

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="update">修改</el-button>

<el-button type="warning" @click="sysRole = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysRoleApi from '@/api/sys-role'

import sysMenuApi from '@/api/sys-menu'

export default {

props: {

// 父组件传递的角色ID

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单对象

sysRole: {},

// 菜单树

menuTree: []

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

created() {

this.getTreeList()

},

methods: {

// 更新角色

update() {

sysRoleApi.update(this.sysRole).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

},

// 根据id查询

getById(id) {

sysRoleApi.get(id).then(res => {

this.sysRole = res.data

// 设置默认选中的权限树

// 获取权限ID集合

const authIds = this.sysRole.authIds

if (authIds && authIds[0]) {

// 手动设置选中

this.$refs.authTree.setCheckedKeys(authIds)

}

})

},

// 树状查询

getTreeList() {

sysMenuApi.getTreeList().then(res => {

this.menuTree = res.data

})

}

}

}

</script>

<style>

</style>

## 6.3 全选bug修改

到这里，我们的角色权限功能大体上开发完毕，但是还存在一些bug，需要我们进行修改。

1. 当用户拥有 **用户管理、角色管理** 的权限时，则必须要能看到 **系统管理** 的权限，因此 **系统管理** 在添加时也需要传到后端
2. 在回显数据时，**系统管理** 的id也被回显出来。但是该菜单是父级菜单，一旦被选中，则下面的子菜单也会全部被选中

总结起来就一句话：

我们只需要查询 **属于指定角色ID的菜单，并且这个菜单的Id不是父级菜单即可**

### 6.3.1 SysMenuController

java

/\*\*

\* 根据角色ID查询选中的菜单

\* 这里不查询父级菜单

\* @param roleId

\* @return

\*/

@RequestMapping(value = "/getRoleSelectMenu/{roleId}", method = RequestMethod.GET)

public Result<List<Long>> getRoleSelectMenu(@PathVariable Long roleId) {

List<Long> ids = sysMenuService.getRoleSelectMenu(roleId);

return new Result<>(ids);

}

### 6.3.2 SysMenuService

java

/\*\*

\* 根据角色ID查询被选中的菜单ID集合

\* @param roleId

\* @return

\*/

List<Long> getRoleSelectMenu(Long roleId);

### 6.3.3 SysMenuServiceImpl

java

@Override

public List<Long> getRoleSelectMenu(Long roleId) {

// 先查出来

List<SysMenu> menuList = sysMenuMapper.getRoleSelectMenu(roleId);

return menuList.stream().map(SysMenu::getMenuId).collect(Collectors.toList());

}

### 6.3.4 SysMenuMapper

java

/\*\*

\* 根据角色ID查询被选中的菜单

\* @param roleId

\* @return

\*/

List<SysMenu> getRoleSelectMenu(Long roleId);

### 6.3.5 SysMenuMapper.xml

xml

<select id="getRoleSelectMenu" resultMap="BaseResultMap">

select m.menu\_id, m.menu\_name, m.parent\_id

from sys\_menu as m

left join sys\_role\_menu srm on m.menu\_id = srm.menu\_id

where m.deleted = 0

and srm.role\_id = #{roleId}

and m.menu\_id not in (

select tm.parent\_id

from sys\_menu tm

inner join sys\_role\_menu s on tm.menu\_id = s.menu\_id

and s.role\_id = #{roleId}

where tm.deleted = 0

)

</select>

### 6.3.6 sysMenu.js

js

getRoleSelectMenu(roleId) { // 查询所有

return request({

url: `/${group\_name}/getRoleSelectMenu/${roleId}`,

method: 'get'

})

},

### 6.3.7 sys-role-update

vue

<template>

<div>

<el-form ref="form" :model="sysRole" label-width="80px" size="small">

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="角色名">

<el-input v-model="sysRole.roleName" placeholder="请输入角色名" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="排序">

<el-input-number v-model="sysRole.roleSort" style="width: 100%" controls-position="right" :min="1" />

</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="权限">

<el-tree

ref="authTree"

:props="{label: 'menuName'}"

:data="menuTree"

show-checkbox

:render-after-expand="false"

node-key="menuId"

/>

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="update">修改</el-button>

<el-button type="warning" @click="sysRole = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysRoleApi from '@/api/sys-role'

import sysMenuApi from '@/api/sys-menu'

export default {

props: {

// 父组件传递的角色ID

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单对象

sysRole: {},

// 菜单树

menuTree: []

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

this.getRoleSelectMenu(newVal)

}

}

},

created() {

this.getTreeList()

},

methods: {

// 更新角色

update() {

const checkKeys = this.$refs.authTree.getCheckedKeys()

const halfCheckKeys = this.$refs.authTree.getHalfCheckedKeys()

// 合并选中的数组和半选中的数组

checkKeys.push(...halfCheckKeys)

this.sysRole.authIds = checkKeys

sysRoleApi.update(this.sysRole).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

},

// 根据id查询

getById(id) {

sysRoleApi.get(id).then(res => {

this.sysRole = res.data

})

},

// 根据角色ID查询出默认选中的数据

getRoleSelectMenu(id) {

sysMenuApi.getRoleSelectMenu(id).then(res => {

const authIds = res.data

if (authIds && authIds[0]) {

// 手动设置选中

this.$refs.authTree.setCheckedKeys(authIds)

} else {

// 如果没有权限信息，就清空默认选中

this.$refs.authTree.setCheckedKeys([])

}

})

},

// 树状查询

getTreeList() {

sysMenuApi.getTreeList().then(res => {

this.menuTree = res.data

})

}

}

}

</script>

<style>

</style>

# 7. 动态路由

到这里，我们已经把基本的RBAC模块编写完毕，接下来我们开始改造路由。

前面我们了解到，vue-element-admin 的路由有两套，一个是 constantRoutes ，代表常量路由，是固定写死在系统中的路由。一个是 asyncRoutes ，代表动态路由，可以由接口去加载的路由。

我们现在所要做的就是把这个路由改为从接口获取。

## 7.1 后端

### 7.1.1 创建 VO

路由的结构比较复杂，因此我们需要创建新的类 RouterVo

java

package com.jg.pochi.pojo.vo;

import lombok.Data;

import java.io.Serializable;

import java.util.List;

/\*\*

\* 路由视图类

\* @Author: 杨德石

\* @Date: 2020/11/16 22:11

\* @Version 1.0

\*/

@Data

public class RouterVo implements Serializable {

/\*\*

\* 路由名称

\*/

private String name;

/\*\*

\* 路由地址

\*/

private String path;

/\*\*

\* 组件地址

\*/

private String component;

/\*\*

\* 当设置noRedirect的时候该路由会在面包屑导航中不能点击

\*/

private String redirect;

/\*\*

\* 是否隐藏

\*/

private boolean hidden;

/\*\*

\* 是否永远展示。如果为true，即使子菜单只有一个，也会展示层级关系

\*/

private boolean alwaysShow;

/\*\*

\* 子菜单

\*/

private List<RouterVo> children;

/\*\*

\* 路由的元信息

\*/

private MetaVo meta;

/\*\*

\* 路由元信息

\*/

@Data

public static class MetaVo implements Serializable {

/\*\*

\* 菜单名称

\*/

private String title;

/\*\*

\* 菜单图标

\*/

private String icon;

public MetaVo(String title, String icon) {

this.title = title;

this.icon = icon;

}

}

}

### 7.1.2 SysMenuController

因为路由实际上是存到菜单表的，所以我们把功能写到菜单的控制层里。

java

/\*\*

\* 获取动态路由

\*

\* @return

\*/

@RequestMapping(value = "/getRouters", method = RequestMethod.GET)

public Result<List<RouterVo>> getRouters() {

List<RouterVo> list = sysMenuService.getRouters();

return new Result<>(list);

}

### 7.1.3 SysMenuService

java

/\*\*

\* 获取动态路由

\* @return

\*/

List<RouterVo> getRouters();

### 7.1.4 SysMenuServiceImpl

构建路由的方法比较复杂，我们这里一点一点讲解

#### getRouters

该方法是构建路由的核心方法，主要功能就是查询出用户所有的菜单，再构造成路由类返回。

该方法的逻辑很简单

1. 查询出当前登录用户所拥有的启用中的所有菜单（权限不要查）
2. 构造成树形结构，也就是 SysMenuVo
3. 构造成路由树

java

@Override

public List<RouterVo> getRouters() {

// 1. 查询出当前登录用户所拥有的启用中的所有菜单（权限不要查）

SysUser loginUser = ShiroUtils.getLoginUser();

List<SysMenu> menuList = sysMenuMapper.getEnableMenuByUserId(loginUser.getId());

// 2. 构造成树形结构，也就是 `SysMenuVo`

List<SysMenuVo> menuVoList = menuList.stream().filter(e -> e.getParentId().equals(CoreConstant.DEFAULT\_PARENT\_ID))

.map(e -> {

// 构造SysMenuVo

SysMenuVo sysMenuVo = new SysMenuVo();

// 拷贝属性

BeanUtils.copyProperties(e, sysMenuVo);

return sysMenuVo;

})

.map(e -> {

// 构造树形结构

e.setChildren(getChildren(e, menuList));

if (e.getChildren() == null) {

// 设置为空集合

e.setChildren(new ArrayList<>(0));

}

return e;

}).collect(Collectors.toList());

// 3. 构造成路由树

return buildMenus(menuVoList);

}

#### buildMenus

查询方法我们先不写，先将构造路由的方法写出来。这个方法的逻辑如下

1. 遍历上面的菜单树
2. 创建 RouterVo 对象，将菜单数据转换成路由视图对象
   1. 构造跳转路径，用多个 '/' 拼接
   2. 构造组件路径，用多个 '/' 拼接
   3. 构造meta数据
3. 如果当前是目录，并且子菜单不为空，就递归构造子菜单

java

/\*\*

\* 构造路由树形结构

\*

\* @param menuVoList

\* @return

\*/

private List<RouterVo> buildMenus(List<SysMenuVo> menuVoList) {

// 1. 遍历上面的菜单树

return menuVoList.stream().map(e -> {

// 2. 创建 `RouterVo` 对象，将菜单数据转换成路由视图对象

RouterVo router = new RouterVo();

router.setHidden(CoreConstant.HIDDEN\_STATE.equals(e.getVisible()));

router.setName(e.getMenuName());

// 2.1. 构造跳转路径，用多个 '/' 拼接

router.setPath(getRouterPath(e));

// 2.2. 构造组件路径，用多个 '/' 拼接

router.setComponent(getComponent(e));

// 2.3. 构造meta数据

router.setMeta(new RouterVo.MetaVo(e.getMenuName(), e.getIcon()));

// 3. 如果当前是目录，并且子菜单不为空，就递归构造子菜单

List<SysMenuVo> children = e.getChildren();

if (!CollectionUtils.isEmpty(children) && StateEnums.FOLDER.getCode().equals(e.getMenuType())) {

router.setAlwaysShow(true);

// 设置redirect

router.setRedirect(CoreConstant.NO\_REDIRECT);

// 递归构造菜单

router.setChildren(buildMenus(children));

} else {

// 子菜单为空的情况下，children不能给null，否则会报错

router.setChildren(new ArrayList<>(0));

}

return router;

}).collect(Collectors.toList());

}

#### getRouterPath

java

/\*\*

\* 构造路由路径

\*

\* @param e

\* @return

\*/

private String getRouterPath(SysMenuVo e) {

if(StateEnums.FOLDER.getCode().equals(e.getMenuType())) {

return CoreConstant.URL\_SPLIT + e.getRouterPath();

}else {

return e.getRouterPath();

}

}

#### getComponent

java

/\*\*

\* 构造组件路径

\*

\* @param e

\* @return

\*/

private String getComponent(SysMenuVo e) {

String component = CoreConstant.DEFAULT\_COMPONENT;

if(StringUtils.isNotEmpty(e.getComponentUrl())) {

component = e.getComponentUrl();

}

return component;

}

### 7.1.5 SysMenuMapper

java

/\*\*

\* 根据用户ID查询启用中的菜单（权限不查）

\* @param id

\* @return

\*/

List<SysMenu> getEnableMenuByUserId(Long id);

### 7.1.6 SysMenuMapper.xml

查询用户菜单的功能需要连的表比较多，不出意外的话这应该是本次课程连表最多的查询。

xml

<select id="getEnableMenuByUserId" resultMap="BaseResultMap">

select \*

from sys\_menu m

inner join sys\_role\_menu srm on m.menu\_id = srm.menu\_id

inner join sys\_role sr on srm.role\_id = sr.role\_id

inner join sys\_user\_role sur on sr.role\_id = sur.role\_id

where m.status = 1

and m.deleted = 0

and m.menu\_type in (1, 2)

and sur.user\_id = #{userId}

order by m.order\_num

</select>

## 7.2 前端

### 7.2.1 API

因为我们新加了接口，所以需要在 sysMenu.js 中配置新的接口地址

js

/\*\*

\* 查询用户的路由

\*/

getRouters() {

return request({

url: `/${groupName}/getRouters`,

method: 'get'

})

}

### 7.2.2 测试

该接口比较复杂，因此我们需要先在任意一个页面中测试一下。

js

getRouters() {

sysMenuApi.getRouters().then((res) => {

console.log(res)

})

},

测试通过，没有问题。接下来就是将我们的路由替换成动态路由了。

### 7.2.3 动态加载路由

我们定位到 srore/modules/permission.js 文件，编写将加载路由的代码加入到这里。

#### 引入Layout

Layout 主区域的布局，所有的组件都基于这个组件。在我们的接口中提供的是字符串的 Layout ，因此我们需要在前端做处理，把layout引入

js

import Layout from '@/layout'

export function filterAsyncRoutes(routes) {

const res = []

routes.forEach(route => {

const tmp = { ...route }

if (tmp.component) {

// Layout组件特殊处理

if (tmp.component === 'Layout') {

tmp.component = Layout

} else {

tmp.component = loadView(route.component)

}

}

if (tmp.children && tmp.children[0]) {

tmp.children = filterAsyncRoutes(tmp.children)

}

res.push(tmp)

})

return res

}

#### 引入组件

在我们接口提供的数据中，并不是完整的引入路径。如果提供了以 @ 开头的完整路径，会报错。因此我们需要手动引入，就是上面代码中的 loadView

js

export const loadView = (view) => { // 路由懒加载

return (resolve) => require([`@/views/${view}`], resolve)

}

#### 加载路由

最后就是我们加载路由了

js

const actions = {

generateRoutes({ commit }) {

return new Promise(resolve => {

sysMenuApi.getRouters().then(res => {

console.log(asyncRoutes, res.data)

const accessedRoutes = filterAsyncRoutes(res.data)

commit('SET\_ROUTES', accessedRoutes)

resolve(accessedRoutes)

})

})

}

}

## 7.3 增加权限

实际上到这里的时候，我们的权限功能基本完毕了，但是我们发现，目前我们的权限只有菜单级别，没有精确到按钮级别，我们需要设置按钮级别的权限。

首先我们将菜单的权限信息完善一下。权限的规范一般按照 大模块:小模块:功能 来定义。这里以用户管理为例，添加 **添加用户、修改用户、删除用户** 三个权限，只给超级管理员分配前两个权限

### 7.3.1 查询用户权限

我们需要在 SysMenuMapper 中提供一个根据用户id查询权限的功能，这里我们查询的就不是权限ID了，而是权限编码。

#### SysUserMapper

java

List<String> getMenuCodeByUserId(Long userId);

#### SysUserMapper.xml

我们这里需要查询权限标识不为空并且用户拥有的权限

xml

<select id="getMenuCodeByUserId" resultType="java.lang.String">

select m.permission

from sys\_menu m

inner join sys\_role\_menu srm on m.menu\_id = srm.menu\_id

inner join sys\_role sr on srm.role\_id = sr.role\_id

inner join sys\_user\_role sur on sr.role\_id = sur.role\_id

where m.status = 1

and m.deleted = 0

and sur.user\_id = #{userId}

and permission is not null

</select>

### 7.3.2 实体类

我们回到 SysUserVo ，现在我们需要给用户Vo里再加上一个权限列表。这里我们只需要存权限的标识符。

java

package com.jg.pochi.pojo.vo;

import com.jg.pochi.pojo.SysRole;

import lombok.Data;

import java.io.Serializable;

import java.util.List;

/\*\*

\* 系统用户视图类

\*

\* @Author: 杨德石

\* @Date: 2020/11/13 21:34

\* @Version 1.0

\*/

@Data

public class SysUserVo implements Serializable {

/\*\*

\* id

\*/

private Long id;

/\*\*

\* 用户名

\*/

private String username;

/\*\*

\* 密码

\*/

private String password;

/\*\*

\* 微信的openid

\*/

private String openId;

/\*\*

\* 邮箱

\*/

private String email;

/\*\*

\* 昵称

\*/

private String nickName;

/\*\*

\* 头像

\*/

private String header;

/\*\*

\* 备注

\*/

private String note;

/\*\*

\* 账号启用状态，1是0否

\*/

private Integer status;

/\*\*

\* 创建时间

\*/

private String createTime;

/\*\*

\* 修改时间

\*/

private String updateTime;

/\*\*

\* 最后登录时间

\*/

private String loginTime;

/\*\*

\* 角色

\*/

private SysRole sysRole;

/\*\*

\* 权限列表

\*/

private List<String> auths;

}

#### realm

在我们的Realm中，处理的就是认证和授权的逻辑。这里需要注意的是，不仅授权逻辑需要查询权限信息，认证逻辑也需要。

**授权逻辑是用来后端鉴权，而认证逻辑的作用是缓存权限以及向前端返回权限**

这里我们就需要将认证功能的SysUser改成Vo类了。

java

package com.jg.pochi.shiro;

import com.jg.pochi.enums.ResultEnums;

import com.jg.pochi.enums.StateEnums;

import com.jg.pochi.exception.PochiException;

import com.jg.pochi.mapper.SysMenuMapper;

import com.jg.pochi.pojo.SysUser;

import com.jg.pochi.pojo.vo.SysUserVo;

import com.jg.pochi.service.SysUserService;

import org.apache.shiro.authc.AuthenticationException;

import org.apache.shiro.authc.AuthenticationInfo;

import org.apache.shiro.authc.AuthenticationToken;

import org.apache.shiro.authc.SimpleAuthenticationInfo;

import org.apache.shiro.authc.UsernamePasswordToken;

import org.apache.shiro.authz.AuthorizationInfo;

import org.apache.shiro.authz.SimpleAuthorizationInfo;

import org.apache.shiro.realm.AuthorizingRealm;

import org.apache.shiro.subject.PrincipalCollection;

import org.springframework.beans.BeanUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import org.springframework.util.CollectionUtils;

import java.util.HashSet;

import java.util.List;

/\*\*

\* 系统用户登录realm

\*

\* @author yds

\*/

@Component("sysUserRealm")

public class SysUserRealm extends AuthorizingRealm {

@Autowired

private SysUserService sysUserService;

@Autowired

private SysMenuMapper sysMenuMapper;

/\*\*

\* 授权方法

\*

\* @param principalCollection

\* @return

\*/

@Override

protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {

// 获取登录用户

SysUserVo sysUserVo = (SysUserVo) principalCollection.getPrimaryPrincipal();

SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();

info.setStringPermissions(new HashSet<>(sysUserVo.getAuths()));

return info;

}

/\*\*

\* 认证

\*

\* @param token

\* @return

\* @throws AuthenticationException

\*/

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {

// 处理登录逻辑

UsernamePasswordToken usernamePasswordToken = (UsernamePasswordToken) token;

String username = usernamePasswordToken.getUsername();

SysUser sysUser = sysUserService.getByUsername(username);

if (sysUser == null) {

throw new PochiException(ResultEnums.LOGIN\_PARAM\_ERROR);

}

if (StateEnums.NOT\_ENABLE.getCode().equals(sysUser.getStatus())) {

// 未启用用户

throw new PochiException(ResultEnums.LOGIN\_PARAM\_ERROR);

}

if (StateEnums.DELETED.getCode().equals(sysUser.getDeleted())) {

// 已删除用户

throw new PochiException(ResultEnums.LOGIN\_PARAM\_ERROR);

}

// 创建SYsUserVo拷贝属性

SysUserVo sysUserVo = new SysUserVo();

BeanUtils.copyProperties(sysUser, sysUserVo);

// 在这里查询权限

List<String> auths = sysMenuMapper.getMenuCodeByUserId(sysUser.getId());

if (CollectionUtils.isEmpty(auths)) {

throw new PochiException("当前用户不具备任何权限，禁止登录");

}

sysUserVo.setAuths(auths);

return new SimpleAuthenticationInfo(sysUserVo, sysUser.getPassword(), this.getName());

}

}

登录这里改完之后，我们需要大改动一些地方。

在我们的部分方法中，使用到了 getLoginUser 方法，并将其返回值强转为 SysUser 。如果不进行修改，则会报出类型转换异常。

接着，就是修改授权方法了。如果不进行授权，则用户可能会跳过页面，直接调用接口，这种情况下权限无法控制。

我们在登录时就已经存了权限信息，因此这里我们可以直接获取登录中用户的权限。

java

/\*\*

\* 授权方法

\*

\* @param principalCollection

\* @return

\*/

@Override

protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {

// 获取登录中的用户

SysUserVo userVo = (SysUserVo) principalCollection.getPrimaryPrincipal();

SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();

info.addStringPermissions(userVo.getAuths());

return info;

}

最后，我们在接口上加上 @RequiresPermissions 注解，这个接口就会进行鉴权了。

编写完毕后，先通过 postman 测试，也可以直接在系统中测试。

#### 全局处理权限不足

我们发现，删除的功能会报操作失败，因为权限不足。因此我们需要全局对该异常进行处理，处理方式和处理自定义异常一样。

java

/\*\*

\* 处理权限不足异常

\* @param exception

\* @return

\*/

@ExceptionHandler(AuthorizationException.class)

public Result<?> authorizationHandler(AuthorizationException exception) {

log.error("权限不足异常处理，", exception);

return new Result<>(ResultEnums.AUTH\_NOT\_FOUNT);

}

### 7.3.3 前端

后端的权限功能编写完成，我们通过系统测试之后发现权限功能正常，但是前端并未把没有权限的按钮隐藏起来。

这里我们可以使用 vue-element-admin 提供的自定义指令 v-permission 。该指令会校验 vuex 中的 roles 是否包含指定的值，因此我们只需要登录成功后将权限信息存储到 roles 即可。

#### 登录

在 vuex 下的 user.js 中我们存储权限

js

// get user info

getInfo({ commit, state }) {

return new Promise((resolve, reject) => {

getInfo().then(response => {

const { data } = response

commit('SET\_USER', data)

if (data.auths) {

commit('SET\_ROLES', data.auths)

} else {

commit('SET\_ROLES', 'admin')

}

resolve(data)

}).catch(error => {

reject(error)

})

})

},

#### 权限控制

之后，在需要控制权限的组件、标签上使用自定义指令即可。多个值之间是 “或” 的关系。

首先注册全局指令。这个指令在 directive/permission 下，在 main.js 中进入

js

import permission from '@/directive/permission'

Vue.use(permission)

接着在需要的地方直接使用即可

html

<div v-permission="['sys:user:add']" class="mid-button-group">

<el-button size="small" type="primary" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

# 8. 日志管理

## 8.1 安装MongoDB

在我们搭建环境的时候，采用了 mysql 存放日志，但是这对数据库的压力较大，因此直接入库不太合适。我们来分析下日志数据的特点。

* 每个请求都需要写入日志，写入负载高
* 日志的请求参数和返回值都需要入表，数据较大
* 日志的数据可能并不需要多高的完整性，偶尔丢几条数据不要紧
* 日志很少会查询，一般只有出了问题的时候才考虑查日志表。

综上，我们发现日志的场景实际上是复合 mongoDB 的解决场景的，因此我们这里安装 mongoDB 来存储日志。

创建目录存放数据、日志

sh

# 进入local

cd /usr/local

# 创建docker目录，以后的挂载文件都放到这里

mkdir -p docker

cd docker

mkdir -p mongodb

# 进入mongodb目录

cd mongodb

# 创建data、log文件夹

mkdir data log

cd data

mkdir db

cd ../

touch log/mongod.log

为目录、文件授权

sh

chmod 777 mongodb

启动

sh

docker run -di --name mongodb --privileged -p 27017:27017 \

-v /usr/local/docker/mongodb/data/db:/data/db \

-v /usr/local/docker/mongodb/conf:/data/configdb \

-v /usr/local/docker/mongodb/log:/data/log \

mongo

## 8.2 引入MongoDB

### 8.2.1 pom.xml

xml

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

### 8.2.2 application.yml

yaml

spring:

data:

mongodb:

host: 39.102.41.53

port: 27017

database: pochi

### 8.2.3 实体类

改造实体类，加上MongoDB的注解

java

package com.jg.pochi.pojo;

import lombok.Data;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.index.Indexed;

import org.springframework.data.mongodb.core.mapping.Document;

import org.springframework.data.mongodb.core.mapping.Field;

import java.io.Serializable;

/\*\*

\* <p>

\* 系统日志，存mongodb实体类

\* </p>

\*

\* @author 杨德石

\* @date 2020-09-22 17:43:33

\* @Version 1.0

\*

\*/

@Data

@Document(collection = "sys\_log")

public class SysLog implements Serializable {

private static final long serialVersionUID = -925216007678145752L;

/\*\*

\* 日志编号

\*/

@Id

private Long logId;

/\*\*

\* 请求路径

\*/

@Field("log\_url")

@Indexed

private String logUrl;

/\*\*

\* 参数

\*/

@Field("log\_params")

private String logParams;

/\*\*

\* 状态，1正常，0异常

\*/

@Field("log\_status")

private Integer logStatus;

/\*\*

\* 异常文本

\*/

@Field("log\_message")

private String logMessage;

/\*\*

\* 浏览器ua标识

\*/

@Field("log\_ua")

private String logUa;

/\*\*

\* controller

\*/

@Field("log\_controller")

private String logController;

/\*\*

\* 请求方式，GET、POST、PUT、DELETE

\*/

@Field("log\_method")

private String logMethod;

/\*\*

\* 响应时间（毫秒）

\*/

@Field("log\_time")

private Long logTime;

/\*\*

\* 返回值

\*/

@Field("log\_result")

private String logResult;

/\*\*

\* 请求ip

\*/

@Field("log\_ip")

private String logIp;

/\*\*

\* 创建时间

\*/

@Field("created\_date")

private String createdDate;

/\*\*

\* 创建人

\*/

@Field("created\_by")

private String createdBy;

}

### SysLogRepository

java

public interface SysLogRepository extends MongoRepository<SysLog, Long> {

}

## 8.3 功能编写

首先清空Service，并移除Mapper（这步也可以不做，不过我们后面不会再用这个了，最好删掉，防止混淆。）

### 8.3.1 SysLogDto

日志涉及到时间区间查询，我们创建个DTO用来接收参数

java

package com.jg.pochi.pojo.dto;

import lombok.Data;

import java.io.Serializable;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/10/1 17:09

\* @Version 1.0

\*/

@Data

public class SysLogDto implements Serializable {

/\*\*

\* 日志编号

\*/

private Long logId;

/\*\*

\* 请求路径

\*/

private String logUrl;

/\*\*

\* 状态，1正常，0异常

\*/

private Integer logStatus;

/\*\*

\* controller

\*/

private String logController;

/\*\*

\* 请求方式，GET、POST、PUT、DELETE

\*/

private String logMethod;

/\*\*

\* 响应时间（毫秒）

\*/

private Long logTime;

/\*\*

\* 请求ip

\*/

private String logIp;

/\*\*

\* 创建时间

\*/

private List<String> createdDate;

/\*\*

\* 创建人

\*/

private String createdBy;

private Integer pageSize;

private Integer pageNumber;

}

### 8.3.2 SysLogController

java

package com.jg.pochi.controller;

import com.jg.pochi.common.Page;

import com.jg.pochi.common.Result;

import com.jg.pochi.pojo.SysLog;

import com.jg.pochi.pojo.dto.SysLogDto;

import com.jg.pochi.service.SysLogService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/17 23:58

\* @Version 1.0

\*/

@RestController

@RequestMapping("/sysLog")

public class SysLogController {

@Autowired

private SysLogService sysLogService;

/\*\*

\* 根据ID查询

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysLog> get(@PathVariable String id) {

SysLog sysLog = sysLogService.get(id);

return new Result<>(sysLog);

}

/\*\*

\* 根据id删除

\* @param id

\* @return

\*/

@RequestMapping(value = "/delete/{id}", method = RequestMethod.DELETE)

public Result<?> delete(@PathVariable String id) {

sysLogService.delete(id);

return new Result<>("删除成功");

}

/\*\*

\* 分页查询

\* @param sysLogDto

\* @return

\*/

@RequestMapping(value = "/getByPage", method = RequestMethod.POST)

public Result<Page<SysLog>> getByPage(@RequestBody SysLogDto sysLogDto) {

Page<SysLog> page = sysLogService.getByPage(sysLogDto);

return new Result<>(page);

}

}

### 8.3.3 SysLogService

java

package com.jg.pochi.service;

import com.jg.pochi.pojo.SysLog;

import com.jg.pochi.pojo.dto.SysLogDto;

import com.jg.pochi.utils.Page;

/\*\*

\* <p>

\* 系统日志，存mongodb服务层接口

\* </p>

\*

\* @author 杨德石

\* @date 2020-09-22 17:43:33

\* @Version 1.0

\*

\*/

public interface SysLogService {

/\*\*

\* 保存

\*

\* @param sysLog

\*/

void save(SysLog sysLog);

/\*\*

\* 根据id查询

\*

\* @param id

\* @return

\*/

SysLog findById(Long id);

/\*\*

\* 框架分页查询

\*

\* @param sysLogDto

\* @return

\*/

Page<SysLog> findAutoByPage(SysLogDto sysLogDto);

/\*\*

\* 根据id删除

\*

\* @param id

\*/

void removeById(Long id);

}

### 8.3.4 SysLogServiceImpl

java

package com.jg.pochi.service.impl;

import com.jg.pochi.dao.SysLogRepository;

import com.jg.pochi.pojo.SysLog;

import com.jg.pochi.pojo.dto.SysLogDto;

import com.jg.pochi.service.SysLogService;

import com.jg.pochi.utils.IdWorker;

import com.jg.pochi.utils.Page;

import com.jg.pochi.utils.StringUtils;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Sort;

import org.springframework.data.mongodb.core.MongoTemplate;

import org.springframework.data.mongodb.core.query.Criteria;

import org.springframework.data.mongodb.core.query.Query;

import org.springframework.stereotype.Service;

import org.springframework.util.CollectionUtils;

import java.util.List;

/\*\*

\* <p>

\* 系统日志，存mongodb服务实现类

\* </p>

\*

\* @author 杨德石

\* @date 2020-09-22 17:43:33

\* @Version 1.0

\*/

@Service

@Slf4j

public class SysLogServiceImpl implements SysLogService {

@Autowired

private SysLogRepository sysLogRepository;

@Autowired

private MongoTemplate mongoTemplate;

@Autowired

private IdWorker idWorker;

/\*\*

\* 保存

\*

\* @param sysLog

\*/

@Override

public void save(SysLog sysLog) {

sysLog.setLogId(idWorker.nextId());

sysLogRepository.save(sysLog);

}

/\*\*

\* 根据id查询

\*

\* @param id

\* @return

\*/

@Override

public SysLog findById(Long id) {

return sysLogRepository.findById(id).get();

}

/\*\*

\* 分页查询

\*

\* @param sysLogDto

\* @return

\*/

@Override

public Page<SysLog> findAutoByPage(SysLogDto sysLogDto) {

// 构造一个查询对象

Query query = new Query();

// 设置参数

if (StringUtils.isNotBlank(sysLogDto.getLogUrl())) {

query.addCriteria(Criteria.where("log\_url").regex("^" + sysLogDto.getLogUrl()));

}

if (sysLogDto.getLogStatus() != null) {

query.addCriteria(Criteria.where("log\_status").is(sysLogDto.getLogStatus()));

}

if (StringUtils.isNotBlank(sysLogDto.getLogController())) {

query.addCriteria(Criteria.where("log\_controller").regex(sysLogDto.getLogController()));

}

if (!CollectionUtils.isEmpty(sysLogDto.getCreatedDate())) {

query.addCriteria(Criteria.where("created\_date").gt(sysLogDto.getCreatedDate().get(0)));

query.addCriteria(Criteria.where("created\_date").lt(sysLogDto.getCreatedDate().get(1)));

}

Integer pageNumber = sysLogDto.getPageNumber();

if (pageNumber == null || pageNumber < 1) {

pageNumber = 1;

sysLogDto.setPageNumber(pageNumber);

}

Integer pageSize = sysLogDto.getPageSize();

if (pageSize == null || pageSize < 1) {

pageSize = 20;

sysLogDto.setPageSize(20);

}

// 跳过多少3条

query.skip((pageNumber - 1) \* pageSize);

// 取出多少条

query.limit(pageSize);

// 构造排序对象

Sort.Order dateOrder = new Sort.Order(Sort.Direction.DESC, "created\_date");

Sort.Order timeOrder = new Sort.Order(Sort.Direction.DESC, "log\_time");

// 设置排序对象

query.with(Sort.by(dateOrder, timeOrder));

List<SysLog> sysLogs = mongoTemplate.find(query, SysLog.class);

long count = mongoTemplate.count(query, SysLog.class);

Page<SysLog> page = new Page<>();

page.setList(sysLogs);

page.setTotalCount((int) count);

page.setTotalPage((int) Math.ceil(count \* 1.0 / pageSize));

return page;

}

/\*\*

\* 根据id删除

\*

\* @param id

\*/

@Override

public void removeById(Long id) {

sysLogRepository.deleteById(id);

}

}

编写完毕后，将报错的地方解决即可。

## 8.4 前端

### 8.4.1 API

在 api 下新建 sysLog.js。

js

import request from '@/utils/request'

var group\_name = 'sysLog'

export default {

getByPage(page) { // 分页查询

return request({

url: `/${group\_name}/getByPage`,

method: 'post',

data: page

})

},

get(id) { // 根据id查询

return request({

url: `/${group\_name}/get/${id}`,

method: 'get'

})

},

deleteById(id) { // 根据id删除

return request({

url: `/${group\_name}/delete`,

method: 'put',

data: { logId: id }

})

}

}

### 8.4.2 sys-log-list

创建 system/log/sys-log-list.vue，内容如下。并通过系统配置路由

vue

<template>

<div>

<!-- 搜索表单开始 -->

<div class="search-form">

<el-form :inline="true" size="small" :model="sysLog">

<el-form-item>

<el-input v-model="sysLog.logUrl" placeholder="请求地址" />

</el-form-item>

<el-form-item>

<el-input v-model="sysLog.logController" placeholder="控制层" />

</el-form-item>

<el-form-item>

<el-select v-model="sysLog.logStatus" clearable placeholder="访问状态 ">

<el-option

label="正常"

value="1"

/>

<el-option

label="异常"

value="2"

/>

</el-select>

</el-form-item>

<el-form-item>

<el-button type="primary" icon="el-icon-search" @click="search">查询</el-button>

<el-button type="warning" icon="el-icon-refresh" @click="sysLog = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

<!-- 搜索表单结束 -->

<!-- 数据表格开始 -->

<div class="data-table">

<el-table

v-loading="loading"

:data="dataPage.list"

header-row-class-name="table-header"

stripe

style="width: 100%"

>

<el-table-column

prop="logUrl"

label="请求地址"

/>

<el-table-column

prop="logStatus"

label="状态"

/>

<el-table-column

prop="logUa"

label="UA"

/>

<el-table-column

prop="logController"

label="控制层"

/>

<el-table-column

prop="logMethod"

label="请求方式"

/>

<el-table-column

prop="logTime"

label="响应时间"

/>

<el-table-column

prop="logIp"

label="请求IP"

/>

<el-table-column

prop="createdDate"

label="创建时间"

/>

<el-table-column

prop="createdBy"

label="创建人"

/>

<el-table-column

label="操作"

>

<template slot-scope="{row}">

<el-button type="text" icon="el-icon-document" @click="toInfo(row.id)">详情</el-button>

<el-dropdown class="button-group">

<span class="el-dropdown-link">

操作<i class="el-icon-arrow-down el-icon--right" />

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item><el-button type="text" icon="el-icon-delete" @click="toDelete(row.logId)">删除</el-button></el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

<el-pagination

class="pagination"

:current-page="sysLog.currentPage"

:page-sizes="[10,20,30,50]"

:page-size="sysLog.pageSize"

layout="total, sizes, prev, pager, next, jumper"

:total="dataPage.totalCount"

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

/>

</div>

<!-- 数据表格结束 -->

<!-- 详情弹窗 -->

<el-dialog title="日志详情" width="30%" :visible.sync="infoDialog">

<sys-log-info :active-id="activeId" />

</el-dialog>

<!-- 详情弹窗结束 -->

</div>

</template>

<script>

import sysLogApi from '@/api/sysLog'

import SysLogInfo from './sys-log-info'

export default {

components: {

SysLogInfo

},

data() {

return {

sysLog: {

currentPage: 1,

pageSize: 10

},

dataPage: {

list: [],

totalCount: 0,

totalPage: 0

},

loading: false,

activeId: '',

infoDialog: false

}

},

created() {

this.getByPage()

},

methods: {

handleSizeChange(val) {

this.sysLog.pageSize = val

this.getByPage()

},

toDelete(id) {

this.$confirm('此操作会删除该日志，是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'danger'

}).then(() => {

sysLogApi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

})

},

toInfo(id) {

this.activeId = id

this.infoDialog = true

},

handleCurrentChange(val) {

this.sysLog.currentPage = val

this.getByPage()

},

search() {

this.sysLog.currentPage = 1

this.getByPage()

},

closeDialog() {

this.infoDialog = false

},

getByPage() {

this.closeDialog()

this.loading = true

sysLogApi.getByPage(this.sysLog).then(res => {

this.dataPage.list = res.data.list

this.dataPage.totalCount = res.data.totalCount

this.loading = false

})

}

}

}

</script>

<style scoped>

.pagination {

margin-top: 15px;

}

.mid-button-group {

margin-bottom: 15px;

}

.button-group {

margin-left: 10px;

}

</style>

### 8.4.3 sys-log-info

vue

<template>

<div>

<el-form v-loading="loading" label-width="80px">

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="控制层">{{ log.logController }}</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="请求IP">{{ log.logIp }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8">

<el-form-item label="请求路径">{{ log.logUrl }}</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="状态 ">{{ log.logStatus }}</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="请求方式">{{ log.logMethod }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8">

<el-form-item label="创建时间">{{ log.createdDate }}</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="创建人">{{ log.createdBy }}</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="响应时间">{{ log.logTime }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="UA标识">{{ log.logUa }}</el-form-item>

</el-col>

</el-row>

<el-row v-if="log.logMessage" :gutter="20">

<el-col :span="24">

<el-form-item label="异常文本">{{ log.logMessage }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="请求参数">{{ log.logParams }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="响应结果">{{ log.logResult }}</el-form-item>

</el-col>

</el-row>

</el-form>

</div>

</template>

<script>

import sysLogApi from '@/api/sysLog'

export default {

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

log: {},

loading: false

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.loading = true

sysLogApi.get(newVal).then(res => {

this.log = res.data

this.loading = false

})

}

}

}

}

</script>

<style>

.header-image {

margin: auto;

width: 100px;

height: 100px;

}

.image-container {

text-align: center;

}

</style>

### 8.4.4 JSON编辑器

日志详情页的请求参数和响应数据有点丑，我们发现，这两个都是JSON格式，因此我们可以使用 vue-element-admin 提供的JSON编辑器功能 。

vue

<template>

<div>

<el-form v-loading="loading" label-width="80px">

<el-row :gutter="20">

<el-col :span="12">

<el-form-item label="控制层">{{ log.logController }}</el-form-item>

</el-col>

<el-col :span="12">

<el-form-item label="请求IP">{{ log.logIp }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8">

<el-form-item label="请求路径">{{ log.logUrl }}</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="状态 ">{{ log.logStatus }}</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="请求方式">{{ log.logMethod }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8">

<el-form-item label="创建时间">{{ log.createdDate }}</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="创建人">{{ log.createdBy }}</el-form-item>

</el-col>

<el-col :span="8">

<el-form-item label="响应时间">{{ log.logTime }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="24">

<el-form-item label="UA标识">{{ log.logUa }}</el-form-item>

</el-col>

</el-row>

<el-row v-if="log.logMessage" :gutter="20">

<el-col :span="24">

<el-form-item label="异常文本">{{ log.logMessage }}</el-form-item>

</el-col>

</el-row>

<el-row :gutter="20" class="json-container">

<el-col :span="24">

<el-form-item label="请求参数"><json-editor v-model="log.logParamsJson" /></el-form-item>

</el-col>

</el-row>

<el-row :gutter="20" class="json-container">

<el-col :span="24">

<el-form-item label="响应结果"><json-editor v-model="log.logResultJson" /></el-form-item>

</el-col>

</el-row>

</el-form>

</div>

</template>

<script>

import sysLogApi from '@/api/sysLog'

import JsonEditor from '@/components/JsonEditor'

export default {

components: { JsonEditor },

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

log: {},

loading: false

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.loading = true

sysLogApi.get(newVal).then(res => {

this.log = res.data

if (res.data.logParams) {

this.log.logParamsJson = JSON.parse(res.data.logParams)

}

if (res.data.logResult) {

this.log.logResultJson = JSON.parse(res.data.logResult)

}

this.loading = false

})

}

}

}

}

</script>

<style>

.header-image {

margin: auto;

width: 100px;

height: 100px;

}

.image-container {

text-align: center;

}

.CodeMirror pre {

line-height: 16px !important;

}

.json-editor {

min-height: 200px;

max-height: 500px;

overflow-y: scroll;

}

</style>

## 8.5 路径排除

最后我们发现，在查询日志时，这个接口的请求也被记录到了日志表，这个是毫无意义的，因此我们需要将其排除。

java

/\*\*

\* 不记录日志的接口

\*/

private static final String[] EXCLUDE\_URLS = {"/sysLog/"};

private boolean exclude(String url) {

for (String excludeUrl : EXCLUDE\_URLS) {

if(url.contains(excludeUrl)) {

return true;

}

}

return false;

}

/\*\*

\* 方法执行之前调用

\*/

@Before("logPointCut()")

public void doBefore(JoinPoint joinPoint) throws Exception {

// 接收到请求，记录请求内容

ServletRequestAttributes attributes = (ServletRequestAttributes) RequestContextHolder.getRequestAttributes();

assert attributes != null;

HttpServletRequest request = attributes.getRequest();

String uri = request.getRequestURI();

if (!exclude(uri)) {

// 记录下请求内容

printRequestLog(joinPoint, request, uri);

}

}

@Around("logPointCut()")

public Object doAround(ProceedingJoinPoint pjp) throws Throwable {

long startTime = System.currentTimeMillis();

Object ob = pjp.proceed();

long time = System.currentTimeMillis() - startTime;

log.info("耗时 : {}", time);

SysLog logger = SystemContext.get().getLogger();

logger.setLogTime(time);

return ob;

}

/\*\*

\* 后置通知

\*

\* @param ret

\*/

@AfterReturning(returning = "ret", pointcut = "logPointCut()")

public void doAfterReturning(Object ret) {

String result = JSON.toJSONString(ret);

log.info("返回值：{}", JSON.toJSONString(ret));

SysLog logger = SystemContext.get().getLogger();

if (!exclude(logger.getLogUrl())) {

logger.setLogResult(result);

sysLogService.save(logger);

}

}

/\*\*

\* 异常通知

\*

\* @param joinPoint

\* @param e

\*/

@AfterThrowing(pointcut = "logPointCut()", throwing = "e")

public void saveExceptionLog(JoinPoint joinPoint, Throwable e) {

SysLog logger = SystemContext.get().getLogger();

if (!exclude(logger.getLogUrl())) {

logger.setLogStatus(StateEnums.REQUEST\_ERROR.getCode());

logger.setLogMessage(e.getMessage());

logger.setLogTime(0L);

sysLogService.save(logger);

}

}

# 9. 阿里云OSS

传统的 fastdfs 社区目前已经不够活跃，而且运维成本较高，文件传输过程中受服务器带宽影响，带宽的升级费用较高，近几年的使用率逐渐下降，开发者开始转向各大云服务器厂商提供的OSS服务。

这里我们推荐阿里云的OSS。

## 9.1 购买OSS

* 登录阿里云ECS控制台（实际上哪个控制台都行，能看所有的阿里云产品就可以 ），点击左上角菜单，找到 **对象存储OSS**

1. ![image-20201001215312833](https://ydsmarkdown.oss-cn-beijing.aliyuncs.com/md/20201001215313.png)

* 点击左侧 **资源包管理**，点击 **购买资源包**
* ![image-20201001215342355](https://ydsmarkdown.oss-cn-beijing.aliyuncs.com/md/20201001215342.png)
* 买40G的就够了，100G的价格直接翻了10倍。

1. ![](https://ydsmarkdown.oss-cn-beijing.aliyuncs.com/md/20201001215829.png)

## 9.2 创建 Bucket

OSS使用bucket来存储文件，一般我们创建多个bucket，用来存放不同类型的文件，便于分组管理。我们这里简单起见就创建一个bucket。只需要选择勾选的内容，其他的都使用默认即可。

![image-20201001221003246](https://ydsmarkdown.oss-cn-beijing.aliyuncs.com/md/20201001221003.png)

## 9.3 配置OSS

鼠标悬浮头像，选择 AcccessKey管理

![image-20201001221625204](https://ydsmarkdown.oss-cn-beijing.aliyuncs.com/md/20201001221646.png)

点击 创建AccessKey，按照步骤操作即可创建

![image-20201001221746632](https://ydsmarkdown.oss-cn-beijing.aliyuncs.com/md/20201001221747.png)

记下AccessKey和Secret

![image-20201001221855980](https://ydsmarkdown.oss-cn-beijing.aliyuncs.com/md/20201001221856.png)

点击进入对应的bucket，点击文件管理，创建一个目录，我们使用目录来管理不同模块下的文件。

我们创建 header、types、product 三个目录，用来存放用户头像、商品分类、商品图片。

到这里，我们的OSS创建完毕。

AccessKey：LTAI4GDrx7BnBXAdbrdNpaT5 AccesSecret：Vh2rVJkkHUxXUiXXzCfML8m6NrQA4L

## 9.4 Java操作OSS

### 9.4.1 引入OSS相关包

xml

<dependency>

<groupId>com.aliyun.oss</groupId>

<artifactId>aliyun-sdk-oss</artifactId>

<version>3.8.0</version>

</dependency>

### 9.4.2 配置OSS

在 application.yml 中配置OSS

yaml

# OSS相关配置信息

aliyun:

oss:

endpoint: http://oss-cn-beijing.aliyuncs.com # oss对外服务的访问域名

access-key-id: LTAI4GEKLMTLzpHj96rKtfM4 # 访问身份验证中用到用户标识

access-key-secret: Zg6piXVfnIkTMv1V5QETNOyAS7jCnL # 用户用于加密签名字符串和oss用来验证签名字符串的密钥

bucket-name: boqii # oss的存储空间

view-url: https://boqii.oss-cn-beijing.aliyuncs.com/ # 预览url

max-size: 10 # 最大文件大小，单位M

upload-dirs: # 上传的目录

- header

- types

- product

image-types: # 上传后缀

- JPG

- PNG

- JPEG

- GIF

创建配置类

java

package com.jg.pochi.oss;

import com.aliyun.oss.OSS;

import com.aliyun.oss.OSSClientBuilder;

import lombok.Data;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Bean;

import org.springframework.stereotype.Component;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/18 21:09

\* @Version 1.0

\*/

@Data

@Component

@ConfigurationProperties(prefix = "aliyun.oss")

public class OssConfig {

private String endpoint;

private String accessKeyId;

private String accessKeySecret;

private String bucketName;

private String viewUrl;

private Integer maxSize;

private List<String> uploadDirs;

private List<String> imageTypes;

@Bean

public OSS oss() {

OSSClientBuilder builder = new OSSClientBuilder();

return builder.build(endpoint, accessKeyId, accessKeySecret);

}

}

### 9.4.3 OssService

java

package com.jg.pochi.service;

import org.springframework.web.multipart.MultipartFile;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/10/1 22:47

\* @Version 1.0

\*/

public interface OssService {

/\*\*

\* 文件上传

\* @param uploadFile

\* @param dir 上传的目录

\* @return

\*/

String upload(MultipartFile uploadFile, String dir);

}

### 9.4.4 OssServiceImpl

java

package com.jg.pochi.oss;

import com.aliyun.oss.OSS;

import com.jg.pochi.enums.ResultEnums;

import com.jg.pochi.exception.PochiException;

import com.jg.pochi.utils.StringUtils;

import lombok.extern.slf4j.Slf4j;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.web.multipart.MultipartFile;

import java.io.ByteArrayInputStream;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.UUID;

/\*\*

\* OSS文件接口默认的实现类

\*

\* @Author: 杨德石

\* @Date: 2020/11/18 21:12

\* @Version 1.0

\*/

@Service

@Slf4j

public class OssServiceImpl implements OssService {

@Autowired

private OssConfig ossConfig;

@Autowired

private OSS oss;

@Override

public String upload(MultipartFile uploadFile, String dir) {

// 校验图片格式

boolean isLegal = false;

String ext = uploadFile.getOriginalFilename();

System.out.println(ext);

for (String imageType : ossConfig.getImageTypes()) {

// 判断如果存在，就把isLegal置为true

if (StringUtils.endsWithIgnoreCase(ext, imageType)) {

isLegal = true;

break;

}

}

if (!isLegal) {

throw new PochiException(ResultEnums.FILE\_TYPE\_ERROR);

}

boolean dirFlag = false;

// 校验文件夹是否存在

for (String uploadDir : ossConfig.getUploadDirs()) {

if (uploadDir.equals(dir)) {

dirFlag = true;

break;

}

}

if (!dirFlag) {

throw new PochiException(ResultEnums.DIR\_NOT\_FOUND);

}

// 校验文件大小

// 计算文件最大大小

long maxSize = ossConfig.getMaxSize() \* 1024 \* 1024L;

if (maxSize < uploadFile.getSize()) {

throw new PochiException(ResultEnums.FILE\_TOO\_LARGE);

}

// 上传文件

// 获取文件名

String filename = getPath(ext, dir);

// 上传文件

try {

oss.putObject(ossConfig.getBucketName(), filename, new ByteArrayInputStream(uploadFile.getBytes()));

} catch (IOException e) {

log.error("文件上传失败，", e);

throw new PochiException("文件上传失败");

}

// 返回文件名

return ossConfig.getViewUrl() + filename;

}

/\*\*

\* 获取文件名

\*

\* @param ext

\* @param dir

\* @return

\*/

private String getPath(String ext, String dir) {

return dir + "/" + new SimpleDateFormat("yyyyMMdd").format(new Date()) + "/"

+ UUID.randomUUID().toString() + "." + ext;

}

}

### 9.4.5 OssTest

java

package com.jg.pochi;

import com.jg.pochi.service.OssService;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.mock.web.MockMultipartFile;

import org.springframework.test.context.junit4.SpringRunner;

import org.springframework.web.multipart.MultipartFile;

import java.io.File;

import java.io.FileInputStream;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/10/1 23:09

\* @Version 1.0

\*/

@RunWith(SpringRunner.class)

@SpringBootTest

public class OssTest {

@Autowired

private OssService ossService;

@Test

public void test() throws Exception {

MultipartFile file = new MockMultipartFile("logo", "png", "", new FileInputStream(new File("H:\\视频\\【SpringBoot+Vue+uniapp】构建波奇喵喵屋商城小程序\\资料\\静态资源\\logo.png")));

String header = ossService.upload(file, "header");

System.out.println(header);

}

}

经测试，图片正常上传，并且可以访问。

接下来就是编写接口了

### 9.4.6 UploadController

java

package com.jg.pochi.controller;

import com.jg.pochi.common.Result;

import com.jg.pochi.oss.OssService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.multipart.MultipartFile;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/18 21:30

\* @Version 1.0

\*/

@RestController

@RequestMapping("/upload")

public class UploadController {

@Autowired

private OssService ossService;

/\*\*

\* 文件上传

\*

\* @param file

\* @param dir

\* @return

\*/

@RequestMapping(value = "/uploadFile", method = RequestMethod.POST)

public Result<String> uploadFile(MultipartFile file, String dir) {

String upload = ossService.upload(file, dir);

return new Result<>("上传成功", upload);

}

}

## 9.5 前端配置

到这里，我们后端的OSS就对接完毕了，接下来就是前端进行对接

### 9.5.1 全局配置上传路径

回忆我们之前学 vue-cli 时，提到过全局配置。即那几个以 .env 开头的文件。我们在这里面配置全局的上传路径

properties

# just a flag

ENV = 'development'

# base api

VUE\_APP\_BASE\_API = '/api'

VUE\_APP\_UPLOAD\_URL = '/api/upload/uploadFile'

这里的配置必须以 VUE\_APP 开头，否则无法正常读取 。

### 9.5.2 使用上传组件上传头像

我们以头像上传为例 ，找到用户添加页，修改代码。

vue

<el-row>

<el-col :span="24">

<el-form-item label="头像">

<el-upload

class="avatar-uploader"

:action="uploadUrl"

:show-file-list="false"

:data="{dir: 'header'}"

:on-success="handleAvatarSuccess"

>

<img v-if="imageUrl" :src="imageUrl" class="avatar">

<i v-else class="el-icon-plus avatar-uploader-icon" />

</el-upload>

</el-form-item>

</el-col>

</el-row>

uploadUrl: process.env.VUE\_APP\_UPLOAD\_URL,

handleAvatarSuccess(res, file) {

this.imageUrl = res.data

this.user.header = this.imageUrl

},

# 10. 轮播图

上面学习完了OSS文件上传，下面我们乘胜追击，做一轮播图内容。

轮播图就是在首页定时切换的大图，虽然我们还没做移动端，但是不影响我们开发功能。

## 10.1 后端

### 10.1.1 SysBanner

java

package com.jg.pochi.pojo;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* 轮播图实体

\*

\* @Author: 杨德石

\* @Date: 2020/11/18 21:55

\* @Version 1.0

\*/

@Data

public class SysBanner implements Serializable {

/\*\*

\* ID

\*/

private Long id;

/\*\*

\* 名称

\*/

private String name;

/\*\*

\* 图片路径

\*/

private String pic;

/\*\*

\* 状态，1启用0弃用

\*/

private Integer status;

/\*\*

\* 点击数

\*/

private Integer clickCount;

/\*\*

\* 下单数

\*/

private Integer orderCount;

/\*\*

\* 链接地址

\*/

private String url;

/\*\*

\* 备注

\*/

private String note;

/\*\*

\* 排序值

\*/

private Integer sort;

/\*\*

\* 创建时间

\*/

private String createTime;

/\*\*

\* 创建人

\*/

private String createBy;

/\*\*

\* 修改时间

\*/

private String updateTime;

/\*\*

\* 修改人

\*/

private String updateBy;

/\*\*

\* 逻辑删除，1是0否

\*/

private Integer deleted;

}

### 10.1.2 SysBannerController

java

package com.jg.pochi.controller;

import com.jg.pochi.common.Page;

import com.jg.pochi.common.Result;

import com.jg.pochi.pojo.SysBanner;

import com.jg.pochi.service.SysBannerService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

/\*\*

\* 轮播图控制器

\*

\* @Author: 杨德石

\* @Date: 2020/11/18 22:00

\* @Version 1.0

\*/

@RestController

@RequestMapping("/sysBanner")

public class SysBannerController {

@Autowired

private SysBannerService sysBannerService;

/\*\*

\* 添加

\*

\* @param sysBanner

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<?> save(@RequestBody SysBanner sysBanner) {

sysBannerService.save(sysBanner);

return new Result<>("添加成功");

}

/\*\*

\* 修改

\*

\* @param sysBanner

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<?> update(@RequestBody SysBanner sysBanner) {

sysBannerService.update(sysBanner);

return new Result<>("修改成功");

}

/\*\*

\* 根据id查询

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysBanner> get(@PathVariable Long id) {

SysBanner sysBanner = sysBannerService.get(id);

return new Result<>(sysBanner);

}

/\*\*

\* 根据id删除

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/delete/{id}", method = RequestMethod.DELETE)

public Result<?> delete(@PathVariable Long id) {

sysBannerService.delete(id);

return new Result<>("删除成功");

}

/\*\*

\* 根据id启用

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/enable/{id}", method = RequestMethod.PUT)

public Result<?> enable(@PathVariable Long id) {

sysBannerService.enable(id);

return new Result<>("启用成功");

}

/\*\*

\* 根据id弃用

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/disable/{id}", method = RequestMethod.PUT)

public Result<?> disable(@PathVariable Long id) {

sysBannerService.disable(id);

return new Result<>("弃用成功");

}

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

@RequestMapping(value = "/getByPage", method = RequestMethod.POST)

public Result<Page<SysBanner>> getByPage(@RequestBody Page<SysBanner> page) {

page = sysBannerService.getByPage(page);

return new Result<>(page);

}

}

### 10.1.3 SysBannerService

java

package com.jg.pochi.service;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysBanner;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/18 22:01

\* @Version 1.0

\*/

public interface SysBannerService {

/\*\*

\* 添加

\*

\* @param sysBanner

\*/

void save(SysBanner sysBanner);

/\*\*

\* 修改

\*

\* @param sysBanner

\*/

void update(SysBanner sysBanner);

/\*\*

\* 根据id查询

\*

\* @param id

\* @return

\*/

SysBanner get(Long id);

/\*\*

\* 根据id删除

\*

\* @param id

\*/

void delete(Long id);

/\*\*

\* 启用

\*

\* @param id

\*/

void enable(Long id);

/\*\*

\* 弃用

\*

\* @param id

\*/

void disable(Long id);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

Page<SysBanner> getByPage(Page<SysBanner> page);

}

### 10.1.4 SysBannerServiceImpl

java

package com.jg.pochi.service.impl;

import com.jg.pochi.common.Page;

import com.jg.pochi.enums.StateEnums;

import com.jg.pochi.mapper.SysBannerMapper;

import com.jg.pochi.pojo.SysBanner;

import com.jg.pochi.pojo.SysUser;

import com.jg.pochi.service.SysBannerService;

import com.jg.pochi.utils.ShiroUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/18 22:01

\* @Version 1.0

\*/

@Service

public class SysBannerServiceImpl implements SysBannerService {

@Autowired

private SysBannerMapper sysBannerMapper;

@Override

public void save(SysBanner sysBanner) {

SysUser loginUser = ShiroUtils.getLoginUser();

sysBanner.setCreateBy(loginUser.getUsername());

sysBanner.setUpdateBy(loginUser.getUsername());

sysBannerMapper.save(sysBanner);

}

@Override

public void update(SysBanner sysBanner) {

SysUser loginUser = ShiroUtils.getLoginUser();

sysBanner.setUpdateBy(loginUser.getUsername());

sysBannerMapper.update(sysBanner);

}

@Override

public SysBanner get(Long id) {

return sysBannerMapper.get(id);

}

@Override

public void delete(Long id) {

sysBannerMapper.delete(id);

}

@Override

@Transactional(rollbackFor = Exception.class)

public void enable(Long id) {

SysBanner banner = sysBannerMapper.get(id);

banner.setStatus(StateEnums.ENABLED.getCode());

sysBannerMapper.update(banner);

}

@Override

public void disable(Long id) {

SysBanner banner = sysBannerMapper.get(id);

banner.setStatus(StateEnums.NOT\_ENABLE.getCode());

sysBannerMapper.update(banner);

}

@Override

public Page<SysBanner> getByPage(Page<SysBanner> page) {

List<SysBanner> list = sysBannerMapper.getByPage(page);

int totalCount = sysBannerMapper.countByPage(page);

page.setList(list);

page.setTotalCount(totalCount);

return page;

}

}

### 10.1.5 SysBannerMapper

java

package com.jg.pochi.mapper;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysBanner;

import org.springframework.stereotype.Component;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 13:46

\* @Version 1.0

\*/

@Component

public interface SysBannerMapper {

/\*\*

\* 保存

\*

\* @param sysBanner

\*/

void save(SysBanner sysBanner);

/\*\*

\* 修改

\*

\* @param sysBanner

\*/

void update(SysBanner sysBanner);

/\*\*

\* 根据id查询

\*

\* @param id

\* @return

\*/

SysBanner get(Long id);

/\*\*

\* 根据id删除

\*

\* @param id

\*/

void delete(Long id);

/\*\*

\* 分页查询

\*

\* @param page

\* @return

\*/

List<SysBanner> getByPage(Page<SysBanner> page);

/\*\*

\* 查询总数

\*

\* @param page

\* @return

\*/

int countByPage(Page<SysBanner> page);

}

### 10.1.6 SysBannerMapper.xml

xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jg.pochi.mapper.SysBannerMapper">

<!-- 通用查询映射结果 -->

<resultMap id="BaseResultMap" type="com.jg.pochi.pojo.SysBanner">

<id column="id" property="id"/>

<result column="name" property="name"/>

<result column="pic" property="pic"/>

<result column="status" property="status"/>

<result column="click\_count" property="clickCount"/>

<result column="order\_count" property="orderCount"/>

<result column="url" property="url"/>

<result column="note" property="note"/>

<result column="sort" property="sort"/>

<result column="create\_time" property="createTime"/>

<result column="create\_by" property="createBy"/>

<result column="update\_time" property="updateTime"/>

<result column="update\_by" property="updateBy"/>

<result column="deleted" property="deleted"/>

</resultMap>

<insert id="save">

insert into sys\_banner(name, pic, status, url, note, sort,

create\_by, update\_by)

values (#{name}, #{pic}, #{status}, #{url}, #{note}, #{sort},

#{createBy}, #{updateBy})

</insert>

<update id="update">

update sys\_banner

<set>

<if test="name!=null and name!=''">

name=#{name},

</if>

<if test="pic!=null and pic!=''">

pic=#{pic},

</if>

<if test="url!=null and url!=''">

url=#{url},

</if>

<if test="note!=null and note!=''">

note=#{note},

</if>

<if test="sort!=null and sort!=''">

sort=#{sort},

</if>

<if test="updateBy!=null and updateBy!=''">

update\_by=#{updateBy}

</if>

</set>

where id = #{id}

</update>

<update id="delete">

update sys\_banner

set deleted = 1

where id = #{id}

</update>

<select id="get" resultMap="BaseResultMap">

select id,

name,

pic,

status,

click\_count,

order\_count,

url,

note,

sort

from sys\_banner

where id = #{id}

</select>

<select id="getByPage" resultMap="BaseResultMap">

select id,

name,

pic,

status,

click\_count,

order\_count,

url,

note,

sort,

create\_time,

update\_time,

create\_by,

update\_by

from sys\_banner

where deleted = 0

<if test="params.status!=null">

and status = #{params.status}

</if>

order by sort asc, create\_time desc

limit #{index}, #{pageSize}

</select>

<select id="countByPage" resultType="java.lang.Integer">

select count(\*)

from sys\_banner

where deleted = 0

<if test="params.status!=null">

and status = #{params.status}

</if>

</select>

</mapper>

## 10.2 前端

### 10.2.1 API

在 api 下创建文件 sysBanner.js。内容如下

js

import request from '@/utils/request'

const groupName = 'sysBanner'

export default {

/\*\*

\* 添加

\*/

save(sysUser) {

return request({

url: `/${groupName}/save`,

method: 'post',

data: sysUser

})

},

/\*\*

\* 修改

\*/

update(sysUser) {

return request({

url: `/${groupName}/update`,

method: 'put',

data: sysUser

})

},

/\*\*

\* 分页

\*/

getByPage(page) {

return request({

url: `/${groupName}/getByPage`,

method: 'post',

data: page

})

},

/\*\*

\* 启用

\*/

enableById(id) {

return request({

url: `/${groupName}/enable/${id}`,

method: 'put'

})

},

/\*\*

\* 禁用

\*/

disableById(id) {

return request({

url: `/${groupName}/disable/${id}`,

method: 'put'

})

},

/\*\*

\* 删除

\*/

deleteById(id) {

return request({

url: `/${groupName}/delete/${id}`,

method: 'delete'

})

},

/\*\*

\* 根据id查询

\*/

get(id) {

return request({

url: `/${groupName}/get/${id}`,

method: 'get'

})

}

}

### 10.2.2 页面编写

在 views/system 下创建目录 banner，并创建 sys-banner-list、sys-banner-add、sys-banner-update 三个文件

#### sys-banner-list

vue

<template>

<div>

<!-- 搜索表单开始 -->

<div class="search-form">

<el-form ref="form" :model="page.params" :inline="true" size="small">

<el-form-item>

<el-select v-model="page.params.status" placeholder="启用状态" clearable>

<el-option label="启用" :value="1" />

<el-option label="禁用" :value="0" />

</el-select>

</el-form-item>

<el-form-item>

<el-button type="primary" icon="el-icon-search" @click="search">查询</el-button>

<el-button type="warning" icon="el-icon-refresh" @click="page.params = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

<!-- 搜索表单结束 -->

<!-- 添加按钮 -->

<div class="button-group">

<el-button type="primary" size="small" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

<!-- 添加按钮结束 -->

<!-- 数据表格 -->

<div class="data-table">

<el-table

header-row-class-name="pochi-table-header"

:data="dataPage.list"

stripe

style="width: 100%"

>

<el-table-column prop="name" label="名称" width="120px" />

<el-table-column prop="pic" label="图片" width="230px" align="center">

<template slot-scope="{row}">

<el-image

:src="row.pic"

fit="fill"

/>

</template>

</el-table-column>

<el-table-column prop="status" label="状态" width="80px">

<template slot-scope="{row}">

<el-switch

v-model="row.status"

:active-value="1"

:inactive-value="0"

@change="changeStatus(row)"

/>

</template>

</el-table-column>

<el-table-column prop="url" label="链接地址" />

<el-table-column prop="clickCount" label="点击数" width="70px" />

<el-table-column prop="orderCount" label="下单数" width="70px" />

<el-table-column prop="sort" label="排序" width="60px" />

<el-table-column prop="createTime" label="创建时间" width="160px" />

<el-table-column prop="createBy" label="创建人" width="110px" />

<el-table-column prop="updateTime" label="修改时间" width="160px" />

<el-table-column prop="updateBy" label="修改人" width="110px" />

<el-table-column label="操作" width="160px">

<template slot-scope="{row}">

<el-button type="text" icon="el-icon-edit" @click="toUpdate(row.id)">修改</el-button>

<el-button type="text" icon="el-icon-delete" @click="toDelete(row.id)">删除</el-button>

</template>

</el-table-column>

</el-table>

</div>

<!-- 分页组件开始 -->

<div class="pageable">

<el-pagination

:current-page="page.pageNumber"

:page-sizes="[10, 20, 30, 50]"

:page-size="10"

background

layout="total, sizes, prev, pager, next, jumper"

:total="dataPage.totalCount"

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

/>

</div>

<!-- 分页组件结束 -->

<!-- 数据表格结束 -->

<!-- 添加弹窗 -->

<el-dialog

title="添加轮播图"

:visible.sync="addDialog"

width="30%"

>

<sys-banner-add @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 添加弹窗结束 -->

<!-- 修改弹窗 -->

<el-dialog

title="修改轮播图"

:visible.sync="updateDialog"

width="30%"

>

<sys-banner-update :active-id="activeId" @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 修改弹窗结束 -->

</div>

</template>

<script>

import sysBannerAPi from '@/api/sys-banner'

import sysBannerAdd from './sys-banner-add'

import sysBannerUpdate from './sys-banner-update'

export default {

components: {

sysBannerAdd,

sysBannerUpdate

},

data() {

return {

// 查询分页对象

page: {

// 查询条件

params: {},

// 当前页

pageNumber: 1,

// 每页条数

pageSize: 10

},

// 控制添加弹窗显示

addDialog: false,

// 控制修改弹窗显示

updateDialog: false,

// 数据分页对象

dataPage: {},

// 当前点击的ID

activeId: ''

}

},

created() {

this.getByPage()

},

methods: {

// 搜索

search() {

this.page.pageNumber = 1

this.getByPage()

},

// 每页条数发生改变触发

handleSizeChange(val) {

this.page.pageSize = val

this.getByPage()

},

// 当前页发生改变时触发

handleCurrentChange(val) {

this.page.pageNumber = val

this.getByPage()

},

// 添加

toAdd() {

this.addDialog = true

},

// 修改

toUpdate(id) {

this.activeId = id

this.updateDialog = true

},

// 改变状态

changeStatus(row) {

if (row.status === 0) {

this.$confirm('是否禁用该图片?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

sysBannerAPi.disableById(row.id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

row.status = 1

})

} else {

this.$confirm('是否启用该图片?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'success'

}).then(() => {

sysBannerAPi.enableById(row.id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

row.status = 0

})

}

},

// 删除

toDelete(id) {

this.$confirm('是否删除该图片?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'error'

}).then(() => {

sysBannerAPi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

})

},

// 关闭弹窗

closeDialog() {

this.addDialog = false

this.updateDialog = false

},

// 分页查询

getByPage() {

if (this.page.params.status === '') {

this.page.params.status = null

}

sysBannerAPi.getByPage(this.page).then(res => {

this.dataPage = res.data

})

}

}

}

</script>

<style>

</style>

#### sys-banner-add

vue

<template>

<div>

<el-form :model="sysBanner" label-width="80px" size="small">

<el-row>

<el-col :span="12" :offset="0">

<el-form-item label="名称">

<el-input v-model="sysBanner.name" placeholder="请输入名称" clearable />

</el-form-item>

</el-col>

<el-col :span="12" :offset="0">

<el-form-item label="排序">

<el-input-number v-model="sysBanner.sort" controls-position="right" style="width: 100%" />

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24" :offset="0">

<el-form-item label="跳转路径">

<el-input v-model="sysBanner.url" placeholder="请输入跳转路径" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24" :offset="0">

<el-form-item label="图片">

<el-upload

class="avatar-uploader"

:action="uploadUrl"

:show-file-list="false"

:data="{dir: 'banner'}"

:headers="{Authorization: token}"

:on-success="handleAvatarSuccess"

>

<img v-if="imageUrl" :src="imageUrl" class="avatar">

<i v-else class="el-icon-plus avatar-uploader-icon" />

</el-upload>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24" :offset="0">

<el-form-item label="备注">

<el-input v-model="sysBanner.note" placeholder="备注" type="textarea" :rows="2" clearable />

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import { mapGetters } from 'vuex'

import sysBannerApi from '@/api/sys-banner'

export default {

data() {

return {

// 表单对象

sysBanner: {},

// 图片上传路径

uploadUrl: process.env.VUE\_APP\_UPLOAD\_URL,

// 用于回显的图片路径

imageUrl: null

}

},

computed: {

...mapGetters([

'token'

])

},

methods: {

// 上传成功的回调

handleAvatarSuccess(res, file) {

this.$message.success(res.msg)

this.imageUrl = res.data

this.sysBanner.pic = this.imageUrl

},

// 添加

add() {

sysBannerApi.save(this.sysBanner).then(res => {

this.$message.success(res.msg)

this.$emit('close')

this.$emit('after')

})

}

}

}

</script>

<style>

.avatar-uploader .el-upload {

border: 1px dashed #d9d9d9;

border-radius: 6px;

cursor: pointer;

position: relative;

overflow: hidden;

}

.avatar-uploader .el-upload:hover {

border-color: #409EFF;

}

.avatar-uploader-icon {

font-size: 28px;

color: #8c939d;

width: 178px;

height: 178px;

line-height: 178px;

text-align: center;

}

.avatar {

width: 178px;

height: 178px;

display: block;

}

</style>

#### sys-banner-update

vue

<template>

<div>

<el-form :model="sysBanner" label-width="80px" size="small">

<el-row>

<el-col :span="12" :offset="0">

<el-form-item label="名称">

<el-input v-model="sysBanner.name" placeholder="请输入名称" clearable />

</el-form-item>

</el-col>

<el-col :span="12" :offset="0">

<el-form-item label="排序">

<el-input-number v-model="sysBanner.sort" controls-position="right" style="width: 100%" />

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24" :offset="0">

<el-form-item label="跳转路径">

<el-input v-model="sysBanner.url" placeholder="请输入跳转路径" clearable />

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24" :offset="0">

<el-form-item label="图片">

<el-upload

class="avatar-uploader"

:action="uploadUrl"

:show-file-list="false"

:data="{dir: 'banner'}"

:headers="{Authorization: token}"

:on-success="handleAvatarSuccess"

>

<img v-if="imageUrl" :src="imageUrl" class="avatar">

<i v-else class="el-icon-plus avatar-uploader-icon" />

</el-upload>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24" :offset="0">

<el-form-item label="备注">

<el-input v-model="sysBanner.note" placeholder="备注" type="textarea" :rows="2" clearable />

</el-form-item>

</el-col>

</el-row>

<el-form-item>

<el-button type="primary" @click="update">修改</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import { mapGetters } from 'vuex'

import sysBannerApi from '@/api/sys-banner'

export default {

props: {

// 父组件传递的轮播图ID

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单对象

sysBanner: {},

// 图片上传路径

uploadUrl: process.env.VUE\_APP\_UPLOAD\_URL,

// 用于回显的图片路径

imageUrl: null

}

},

computed: {

...mapGetters([

'token'

])

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

methods: {

// 上传成功的回调

handleAvatarSuccess(res, file) {

this.$message.success(res.msg)

this.imageUrl = res.data

this.sysBanner.pic = this.imageUrl

},

// 根据ID查询

getById(id) {

sysBannerApi.get(id).then(res => {

this.sysBanner = res.data

this.imageUrl = this.sysBanner.pic

})

},

// 修改

update() {

sysBannerApi.update(this.sysBanner).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

}

}

}

</script>

<style>

.avatar-uploader .el-upload {

border: 1px dashed #d9d9d9;

border-radius: 6px;

cursor: pointer;

position: relative;

overflow: hidden;

}

.avatar-uploader .el-upload:hover {

border-color: #409EFF;

}

.avatar-uploader-icon {

font-size: 28px;

color: #8c939d;

width: 300px;

height: 150px;

line-height: 150px;

text-align: center;

}

.avatar {

width: 300px;

height: 150px;

display: block;

}

</style>

# 11. 通知公告

通知公告功能也是用于移动端首页展示，用来通知用户，功能比较单一，我们快速写完CRUD

## 11.1 后端

### 11.1.1 SysNotice

java

package com.jg.pochi.pojo;

import com.jg.pochi.framework.annotation.mybatis.Column;

import com.jg.pochi.framework.annotation.mybatis.GenerationType;

import com.jg.pochi.framework.annotation.mybatis.Id;

import com.jg.pochi.framework.annotation.mybatis.Table;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* <p>

\* 公告实体类

\* </p>

\*

\* @author 杨德石

\* @date 2020-09-22 17:43:33

\* @Version 1.0

\*

\*/

@Data

@Table(name = "sys\_notice")

public class SysNotice implements Serializable {

private static final long serialVersionUID = 563816007678152635L;

@Id(strategy = GenerationType.IDENTITY)

@Column(name = "notice\_id")

private Long noticeId;

/\*\*

\* 公告标题

\*/

@Column(name = "notice\_title")

private String noticeTitle;

/\*\*

\* 公告内容

\*/

@Column(name = "notice\_content")

private String noticeContent;

/\*\*

\* 排序值

\*/

@Column(name = "sorted")

private Integer sorted;

@Column(name = "created\_by")

private String createdBy;

@Column(name = "created\_time")

private String createdTime;

/\*\*

\* 是否删除，1是0否

\*/

@Column(name = "deleted")

private Integer deleted;

/\*\*

\* 是否启用，1是0否

\*/

@Column(name = "enabled")

private Integer enabled;

}

### 11.1.2 SysNoticeController

java

package com.jg.pochi.controller;

import com.jg.pochi.common.Page;

import com.jg.pochi.common.Result;

import com.jg.pochi.pojo.SysNotice;

import com.jg.pochi.service.SysNoticeService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/20 21:11

\* @Version 1.0

\*/

@RestController

@RequestMapping("/sysNotice")

public class SysNoticeController {

@Autowired

private SysNoticeService sysNoticeService;

/\*\*

\* 添加

\*

\* @param sysNotice

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<?> save(@RequestBody SysNotice sysNotice) {

sysNoticeService.save(sysNotice);

return new Result<>("添加成功");

}

/\*\*

\* 修改

\*

\* @param sysNotice

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<?> update(@RequestBody SysNotice sysNotice) {

sysNoticeService.update(sysNotice);

return new Result<>("修改成功");

}

/\*\*

\* 根据id查询

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysNotice> get(@PathVariable Long id) {

SysNotice sysNotice = sysNoticeService.get(id);

return new Result<>(sysNotice);

}

/\*\*

\* 根据id删除

\*

\* @param id

\* @return

\*/

@RequestMapping(value = "/delete/{id}", method = RequestMethod.DELETE)

public Result<?> delete(@PathVariable Long id) {

sysNoticeService.delete(id);

return new Result<>("删除成功");

}

/\*\*

\* 启用

\* @param id

\* @return

\*/

@RequestMapping(value = "/enable/{id}", method = RequestMethod.PUT)

public Result<?> enable(@PathVariable Long id) {

sysNoticeService.enable(id);

return new Result<>("启用成功");

}

/\*\*

\* 禁用

\* @param id

\* @return

\*/

@RequestMapping(value = "/disable/{id}", method = RequestMethod.PUT)

public Result<?> disable(@PathVariable Long id) {

sysNoticeService.disable(id);

return new Result<>("禁用成功");

}

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

@RequestMapping(value = "/getByPage", method = RequestMethod.POST)

public Result<Page<SysNotice>> getByPage(@RequestBody Page<SysNotice> page) {

page = sysNoticeService.getByPage(page);

return new Result<>(page);

}

}

### 11.1.3 SysNoticeService

java

package com.jg.pochi.service;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysNotice;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/20 21:11

\* @Version 1.0

\*/

public interface SysNoticeService {

/\*\*

\* 添加

\* @param sysNotice

\*/

void save(SysNotice sysNotice);

/\*\*

\* 修改

\* @param sysNotice

\*/

void update(SysNotice sysNotice);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysNotice get(Long id);

/\*\*

\* 根据id删除

\* @param id

\*/

void delete(Long id);

/\*\*

\* 启用

\* @param id

\*/

void enable(Long id);

/\*\*

\* 禁用

\* @param id

\*/

void disable(Long id);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

Page<SysNotice> getByPage(Page<SysNotice> page);

}

### 11.1.4 SysNoticeServiceImpl

java

package com.jg.pochi.service.impl;

import com.jg.pochi.common.Page;

import com.jg.pochi.enums.StateEnums;

import com.jg.pochi.mapper.SysNoticeMapper;

import com.jg.pochi.pojo.SysNotice;

import com.jg.pochi.pojo.vo.SysUserVo;

import com.jg.pochi.service.SysNoticeService;

import com.jg.pochi.utils.ShiroUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/20 21:11

\* @Version 1.0

\*/

@Service

public class SysNoticeServiceImpl implements SysNoticeService {

@Autowired

private SysNoticeMapper sysNoticeMapper;

@Override

public void save(SysNotice sysNotice) {

// 创建人默认值

SysUserVo loginUser = ShiroUtils.getLoginUser();

sysNotice.setCreatedBy(loginUser.getUsername());

sysNoticeMapper.save(sysNotice);

}

@Override

public void update(SysNotice sysNotice) {

sysNoticeMapper.update(sysNotice);

}

@Override

public SysNotice get(Long id) {

return sysNoticeMapper.get(id);

}

@Override

public void delete(Long id) {

sysNoticeMapper.delete(id);

}

@Override

public void enable(Long id) {

SysNotice sysNotice = sysNoticeMapper.get(id);

sysNotice.setEnabled(StateEnums.ENABLED.getCode());

sysNoticeMapper.updateEnable(sysNotice);

}

@Override

public void disable(Long id) {

SysNotice sysNotice = sysNoticeMapper.get(id);

sysNotice.setEnabled(StateEnums.NOT\_ENABLE.getCode());

sysNoticeMapper.updateEnable(sysNotice);

}

@Override

public Page<SysNotice> getByPage(Page<SysNotice> page) {

List<SysNotice> list = sysNoticeMapper.getByPage(page);

Integer totalCount = sysNoticeMapper.countByPage(page);

page.setList(list);

page.setTotalCount(totalCount);

return page;

}

}

### 11.1.5 SysNoticeMapper

java

package com.jg.pochi.mapper;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysNotice;

import org.springframework.stereotype.Component;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 13:46

\* @Version 1.0

\*/

@Component

public interface SysNoticeMapper {

/\*\*

\* 添加

\* @param sysNotice

\*/

void save(SysNotice sysNotice);

/\*\*

\* 修改

\* @param sysNotice

\*/

void update(SysNotice sysNotice);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysNotice get(Long id);

/\*\*

\* 根据id删除

\* @param id

\*/

void delete(Long id);

/\*\*

\* 更新启用状态

\* @param sysNotice

\*/

void updateEnable(SysNotice sysNotice);

/\*\*

\* 查询总数

\* @param page

\* @return

\*/

Integer countByPage(Page<SysNotice> page);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

List<SysNotice> getByPage(Page<SysNotice> page);

}

### 11.1.6 SysNoticeMapper.xml

xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jg.pochi.mapper.SysNoticeMapper">

<!-- 通用查询映射结果 -->

<resultMap id="BaseResultMap" type="com.jg.pochi.pojo.SysNotice">

<id column="notice\_id" property="noticeId"/>

<result column="notice\_title" property="noticeTitle"/>

<result column="notice\_content" property="noticeContent"/>

<result column="sorted" property="sorted"/>

<result column="created\_by" property="createdBy"/>

<result column="created\_time" property="createdTime"/>

<result column="delete" property="delete"/>

<result column="enabled" property="enabled"/>

</resultMap>

<insert id="save">

insert into sys\_notice(notice\_title, notice\_content, sorted, created\_by)

VALUES (#{noticeTitle}, #{noticeContent}, #{sorted}, #{createdBy})

</insert>

<update id="update">

update sys\_notice

<set>

<if test="noticeTitle!=null and noticeTitle!=''">

notice\_title = #{noticeTitle},

</if>

<if test="noticeContent!=null and noticeContent!=''">

notice\_content = #{noticeContent},

</if>

<if test="sorted!=null">

sorted = #{sorted},

</if>

</set>

where notice\_id = #{noticeId}

</update>

<update id="delete">

update sys\_notice

set deleted = 1

where notice\_id = #{noticeId}

</update>

<update id="updateEnable">

update sys\_notice

set enabled = #{enabled}

where notice\_id = #{noticeId}

</update>

<select id="get" resultMap="BaseResultMap">

select notice\_id,

notice\_title,

notice\_content,

sorted,

created\_by,

created\_time

from sys\_notice

where notice\_id = #{noticeId}

</select>

<select id="getByPage" resultMap="BaseResultMap">

select notice\_id,

notice\_title,

sorted,

created\_by,

created\_time,

enabled

from sys\_notice

where deleted = 0

<if test="params.noticeTitle!=null and params.noticeTitle!=null">

and notice\_title = #{params.noticeTitle}

</if>

<if test="params.enabled!=null and params.enabled!=null">

and enabled = #{params.enabled}

</if>

order by sorted

limit #{index}, #{pageSize}

</select>

<select id="countByPage" resultType="java.lang.Integer">

select count(\*)

from sys\_notice

where deleted = 0

<if test="params.noticeTitle!=null and params.noticeTitle!=null">

and notice\_title = #{params.noticeTitle}

</if>

<if test="params.enabled!=null and params.enabled!=null">

and enabled = #{params.enabled}

</if>

</select>

</mapper>

## 11.2 前端

### 11.2.1 API

在 api 下创建 sysNotice.js，内容如下。

js

import request from '@/utils/request'

var group\_name = 'sysNotice'

export default {

getByPage(page) { // 分页查询

return request({

url: `/${group\_name}/getByPage`,

method: 'post',

data: page

})

},

save(sysNotice) { // 保存

return request({

url: `/${group\_name}/save`,

method: 'post',

data: sysNotice

})

},

get(id) { // 根据id查询

return request({

url: `/${group\_name}/get/${id}`,

method: 'get'

})

},

update(sysNotice) { // 更新

return request({

url: `/${group\_name}/update`,

method: 'put',

data: sysNotice

})

},

deleteById(id) { // 根据id删除

return request({

url: `/${group\_name}/delete`,

method: 'put',

data: { noticeId: id }

})

},

enableById(id) { // 启用

return request({

url: `/${group\_name}/enable`,

method: 'put',

data: {

noticeId: id

}

})

},

disableById(id) { // 禁用

return request({

url: `/${group\_name}/disable`,

method: 'put',

data: {

noticeId: id

}

})

}

}

### 11.2.2 页面编写

在 views/system 下创建 notice 目录，并分别创建 sys-notice-list.vue、sys-notice-add.vue、sys-notice-update.vue、sys-notice-info.vue，接着在系统中配置路由。

#### sys-notice-list

vue

<template>

<div>

<!-- 搜索表单开始 -->

<div class="search-form">

<el-form :inline="true" size="small" :model="page.params">

<el-form-item>

<el-input v-model="page.params.like\_noticeTitle" placeholder="标题" />

</el-form-item>

<el-form-item>

<el-select v-model="page.params.eq\_enabled" clearable placeholder="启用状态">

<el-option

label="启用"

:value="1"

/>

<el-option

label="禁用"

:value="0"

/>

</el-select>

</el-form-item>

<el-form-item>

<el-button type="primary" icon="el-icon-search" @click="search">查询</el-button>

<el-button type="warning" icon="el-icon-refresh" @click="page.params = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

<!-- 搜索表单结束 -->

<!-- 菜单组开始 -->

<div class="mid-button-group">

<el-button size="small" type="primary" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

<!-- 菜单组结束 -->

<!-- 数据表格开始 -->

<div class="data-table">

<el-table

v-loading="loading"

:data="dataPage.list"

header-row-class-name="table-header"

stripe

style="width: 100%"

>

<el-table-column

prop="noticeTitle"

label="标题"

/>

<el-table-column

prop="sorted"

label="排序值"

/>

<el-table-column

prop="createdBy"

label="创建人"

/>

<el-table-column

prop="createdTime"

label="创建时间"

/>

<el-table-column

prop="enabled"

label="启用状态"

/>

<el-table-column

label="操作"

>

<template slot-scope="{row}">

<el-button type="text" icon="el-icon-document" @click="toInfo(row.noticeId)">详情</el-button>

<el-dropdown class="button-group">

<span class="el-dropdown-link">

操作<i class="el-icon-arrow-down el-icon--right" />

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item><el-button type="text" icon="el-icon-edit" @click="toUpdate(row.noticeId)">修改</el-button></el-dropdown-item>

<el-dropdown-item><el-button type="text" icon="el-icon-delete" @click="toDelete(row.noticeId)">删除</el-button></el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

<el-pagination

class="pagination"

:current-page="page.currentPage"

:page-sizes="[10,20,30,50]"

:page-size="page.pageSize"

layout="total, sizes, prev, pager, next, jumper"

:total="dataPage.totalCount"

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

/>

</div>

<!-- 数据表格结束 -->

<!-- 添加弹窗 -->

<el-dialog title="添加通知" width="60%" :visible.sync="addDialog">

<sys-notice-add @after="getByPage" />

</el-dialog>

<!-- 添加弹窗结束 -->

<!-- 修改弹窗 -->

<el-dialog title="修改通知" width="60%" :visible.sync="updateDialog">

<sys-notice-update :active-id="activeId" @after="getByPage" />

</el-dialog>

<!-- 修改弹窗结束 -->

<!-- 详情弹窗 -->

<el-dialog title="通知详情" width="60%" :visible.sync="infoDialog">

<sys-notice-info :active-id="activeId" />

</el-dialog>

<!-- 详情弹窗结束 -->

</div>

</template>

<script>

import sysNoticeApi from '@/api/sysNotice'

import SysNoticeAdd from './sys-notice-add'

import SysNoticeUpdate from './sys-notice-update'

import SysNoticeInfo from './sys-notice-info'

export default {

components: {

SysNoticeAdd,

SysNoticeUpdate,

SysNoticeInfo

},

data() {

return {

page: {

currentPage: 1,

pageSize: 10,

params: {}

},

dataPage: {

list: [],

totalCount: 0,

totalPage: 0

},

addDialog: false,

loading: false,

activeId: '',

updateDialog: false,

infoDialog: false

}

},

created() {

this.getByPage()

},

methods: {

handleSizeChange(val) {

this.page.pageSize = val

this.getByPage()

},

toAdd() {

this.addDialog = true

},

toUpdate(id) {

this.activeId = id

this.updateDialog = true

},

toDelete(id) {

this.$confirm('此操作会删除该通知，是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'danger'

}).then(() => {

sysNoticeApi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

})

},

changeStatus(row) {

if (row.status === 1) {

this.toEnable(row)

} else {

this.toDisable(row)

}

},

toEnable(row) {

this.$confirm('此操作会启用该通知，是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'success'

}).then(() => {

sysNoticeApi.enableById(row.noticeId).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

row.status = 0

})

},

toDisable(row) {

this.$confirm('此操作会禁用该通知，是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

sysNoticeApi.disableById(row.noticeId).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

}).catch(() => {

row.status = 1

})

},

toInfo(id) {

this.activeId = id

this.infoDialog = true

},

handleCurrentChange(val) {

this.page.currentPage = val

this.getByPage()

},

search() {

this.page.currentPage = 1

this.getByPage()

},

closeDialog() {

this.addDialog = false

this.updateDialog = false

this.infoDialog = false

},

getByPage() {

this.closeDialog()

this.loading = true

sysNoticeApi.getByPage(this.page).then(res => {

this.dataPage.list = res.data.list

this.dataPage.totalCount = res.data.totalCount

this.dataPage.totalPage = res.data.totalPage

this.loading = false

})

}

}

}

</script>

<style scoped>

.pagination {

margin-top: 15px;

}

.mid-button-group {

margin-bottom: 15px;

}

.button-group {

margin-left: 10px;

}

</style>

#### sys-notice-add

vue

<template>

<div>

<el-form size="small" label-width="80px">

<el-row>

<el-col :span="24">

<el-form-item label="标题">

<el-input

v-model="notice.noticeTitle"

placeholder="请输入标题"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24">

<el-form-item label="排序值">

<el-input-number

v-model="notice.sorted"

controls-position="right"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-form-item label="内容">

<markdown-editor ref="markdownEditor" v-model="notice.noticeContent" language="zh\_CN" height="500px" />

</el-form-item>

</el-row>

<el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="notice = {}">重置</el-button>

</el-form-item>

</el-row>

</el-form>

</div>

</template>

<script>

import MarkdownEditor from '@/components/MarkdownEditor'

import sysNoticeApi from '@/api/sysNotice'

export default {

components: { MarkdownEditor },

data() {

return {

notice: {

}

}

},

methods: {

add() {

if (this.notice.noticeContent) {

this.notice.noticeContent = this.$refs.markdownEditor.getHtml()

}

sysNoticeApi.save(this.notice).then(res => {

this.$message.success(res.msg)

this.$emit('after')

})

}

}

}

</script>

#### sys-notice-update

vue

<template>

<div>

<el-form size="small" label-width="80px">

<el-row>

<el-col :span="24">

<el-form-item label="标题">

<el-input

v-model="notice.noticeTitle"

placeholder="请输入标题"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24">

<el-form-item label="排序值">

<el-input-number

v-model="notice.sorted"

controls-position="right"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-form-item label="内容">

<markdown-editor ref="markdownEditor" v-model="content" language="zh\_CN" height="500px" />

</el-form-item>

</el-row>

<el-row>

<el-form-item>

<el-button type="primary" @click="update">添加</el-button>

<el-button type="warning" @click="notice = {}">重置</el-button>

</el-form-item>

</el-row>

</el-form>

</div>

</template>

<script>

import MarkdownEditor from '@/components/MarkdownEditor'

import sysNoticeApi from '@/api/sysNotice'

export default {

components: { MarkdownEditor },

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

notice: {

},

content: '',

loading: false

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.loading = true

sysNoticeApi.get(newVal).then(res => {

this.notice = res.data

if (this.notice.noticeContent) {

this.$refs.markdownEditor.setHtml(this.notice.noticeContent)

}

this.loading = false

})

}

}

},

methods: {

update() {

if (this.content) {

this.notice.noticeContent = this.$refs.markdownEditor.getHtml()

}

sysNoticeApi.update(this.notice).then(res => {

this.$message.success(res.msg)

this.$emit('after')

})

}

}

}

</script>

#### sys-notice-info

vue

<template>

<div>

<el-form v-loading="loading" label-width="80px">

<el-row>

<el-col :span="24">

<el-form-item label="标题">

{{ notice.noticeTitle }}

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24">

<el-form-item label="排序值">

{{ notice.sorted }}

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="12">

<el-form-item label="创建人">

{{ notice.createdBy }}

</el-form-item>

</el-col>

<el-col :span="24">

<el-form-item label="创建时间">

{{ notice.createdTime }}

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-form-item label="内容">

{{ notice.noticeContent }}

</el-form-item>

</el-row>

</el-form>

</div>

</template>

<script>

import sysNoticeApi from '@/api/sysNotice'

export default {

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

notice: {},

loading: false

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.loading = true

sysNoticeApi.get(newVal).then(res => {

this.notice = res.data

this.loading = false

})

}

}

}

}

</script>

### 11.2.3 Markdown编辑器

通知功能中涉及到了通知内容的编写。ElementUI提供的文本域不够强大，我们需要更加强大的编辑器，这里有两个选择，一个是 **富文本编辑器**，一个是 **Markdown** 编辑器。本课程两个都用，通知公告模块采用Markdown。

#### 使用

我们参考 views/components-demo/markdown.vue 下的代码，vue-element-admin 已经为我们封装好了 markdown 编辑器，直接引入使用即可。

markdown编辑器直接获取内容是markdown语法，不便于我们在页面中渲染，因此提供了 getHtml 方法转成 html。同样也提供了 setHtml 方法将html转为markdown

#### sys-notice-add

vue

<template>

<div>

<el-form size="small" label-width="80px">

<el-row>

<el-col :span="24">

<el-form-item label="标题">

<el-input

v-model="notice.noticeTitle"

placeholder="请输入标题"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24">

<el-form-item label="排序值">

<el-input-number

v-model="notice.sorted"

controls-position="right"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-form-item label="内容">

<markdown-editor ref="markdownEditor" v-model="notice.noticeContent" language="zh\_CN" height="500px" />

</el-form-item>

</el-row>

<el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="notice = {}">重置</el-button>

</el-form-item>

</el-row>

</el-form>

</div>

</template>

<script>

import MarkdownEditor from '@/components/MarkdownEditor'

import sysNoticeApi from '@/api/sysNotice'

export default {

components: { MarkdownEditor },

data() {

return {

notice: {

}

}

},

methods: {

add() {

if (this.notice.noticeContent) {

this.notice.noticeContent = this.$refs.markdownEditor.getHtml()

}

sysNoticeApi.save(this.notice).then(res => {

this.$message.success(res.msg)

this.$emit('after')

})

}

}

}

</script>

这里使用的是 tui.editor。这个编辑器默认预览页有点问题，虽然不影响添加，但是看着不好看，我们需要修改这个bug。

#### markdown编辑器bug解决

打开node\_modules包中的tui-editor文件夹，在js文件中搜索 containerTmpl 变量，替换成下面内容

js

var containerTmpl = ['<div class="tui-editor">', '<div class="te-md-container">', '<div class="te-editor">', '</div>', '<div class="te-md-splitter" >', '</div>', '<div class="te-preview">', '</div>', '</div>', '<div class="te-ww-container">', '<div class="te-editor">', '</div>', '</div>', '</div>'].join('');

#### sys-notice-update

我们在添加页把markdown内容转成了html，在修改页要先转回去

vue

<template>

<div>

<el-form size="small" label-width="80px">

<el-row>

<el-col :span="24">

<el-form-item label="标题">

<el-input

v-model="notice.noticeTitle"

placeholder="请输入标题"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24">

<el-form-item label="排序值">

<el-input-number

v-model="notice.sorted"

controls-position="right"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-form-item label="内容">

<markdown-editor ref="markdownEditor" v-model="content" language="zh\_CN" height="500px" />

</el-form-item>

</el-row>

<el-row>

<el-form-item>

<el-button type="primary" @click="update">添加</el-button>

<el-button type="warning" @click="notice = {}">重置</el-button>

</el-form-item>

</el-row>

</el-form>

</div>

</template>

<script>

import MarkdownEditor from '@/components/MarkdownEditor'

import sysNoticeApi from '@/api/sysNotice'

export default {

components: { MarkdownEditor },

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

notice: {

},

content: '',

loading: false

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.loading = true

sysNoticeApi.get(newVal).then(res => {

this.notice = res.data

if (this.notice.noticeContent) {

this.$refs.markdownEditor.setHtml(this.notice.noticeContent)

}

this.loading = false

})

}

}

},

methods: {

update() {

if (this.content) {

this.notice.noticeContent = this.$refs.markdownEditor.getHtml()

}

sysNoticeApi.update(this.notice).then(res => {

this.$message.success(res.msg)

this.$emit('after')

})

}

}

}

</script>

#### sys-notice-info

改成HTML之后，在info页也需要展示HTML

vue

<template>

<div>

<el-form v-loading="loading" label-width="80px">

<el-row>

<el-col :span="24">

<el-form-item label="标题">

{{ notice.noticeTitle }}

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="24">

<el-form-item label="排序值">

{{ notice.sorted }}

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-col :span="12">

<el-form-item label="创建人">

{{ notice.createdBy }}

</el-form-item>

</el-col>

<el-col :span="24">

<el-form-item label="创建时间">

{{ notice.createdTime }}

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-form-item label="内容">

<span v-html="notice.noticeContent" />

</el-form-item>

</el-row>

</el-form>

</div>

</template>

<script>

import sysNoticeApi from '@/api/sysNotice'

export default {

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

notice: {},

loading: false

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.loading = true

sysNoticeApi.get(newVal).then(res => {

this.notice = res.data

this.loading = false

})

}

}

}

}

</script>

#### 图片上传

默认情况下，tui.editor 上传图片是将图片转成base64。这样比较占用数据库资源，而且查询和传输都比较慢，我们需要自己定制图片上传。

以下内容直接复制即可，不需要理解。

* 找到 components/MarkdownEditor，打开 default-options.js，注释掉图片组件。
* 打开 index.vue，替换成下面的内容

1. vue
2. <template>
3. <div>
4. <div :id="id" />
5. <input ref="files" style="display: none" type="file" accept="image/\*" @change="uploadFile">
6. </div>
7. </template>
8. <script>
9. // deps for editor
10. import 'codemirror/lib/codemirror.css' // codemirror
11. import 'tui-editor/dist/tui-editor.css' // editor ui
12. import 'tui-editor/dist/tui-editor-contents.css' // editor content
13. import Editor from 'tui-editor'
14. import defaultOptions from './default-options'
15. import axios from 'axios'
16. export default {
17. name: 'MarddownEditor',
18. props: {
19. value: {
20. type: String,
21. default: ''
22. },
23. id: {
24. type: String,
25. required: false,
26. default() {
27. return 'markdown-editor-' + +new Date() + ((Math.random() \* 1000).toFixed(0) + '')
28. }
29. },
30. options: {
31. type: Object,
32. default() {
33. return defaultOptions
34. }
35. },
36. mode: {
37. type: String,
38. default: 'markdown'
39. },
40. height: {
41. type: String,
42. required: false,
43. default: '300px'
44. },
45. language: {
46. type: String,
47. required: false,
48. default: 'en\_US' // https://github.com/nhnent/tui.editor/tree/master/src/js/langs
49. }
50. },
51. data() {
52. return {
53. editor: null,
54. token: ''
55. }
56. },
57. computed: {
58. editorOptions() {
59. const options = Object.assign({}, defaultOptions, this.options)
60. options.initialEditType = this.mode
61. options.height = this.height
62. options.language = this.language
63. return options
64. }
65. },
66. watch: {
67. value(newValue, preValue) {
68. if (newValue !== preValue && newValue !== this.editor.getValue()) {
69. this.editor.setValue(newValue)
70. }
71. },
72. language(val) {
73. this.destroyEditor()
74. this.initEditor()
75. },
76. height(newValue) {
77. this.editor.height(newValue)
78. },
79. mode(newValue) {
80. this.editor.changeMode(newValue)
81. }
82. },
83. mounted() {
84. this.initEditor()
85. },
86. destroyed() {
87. this.destroyEditor()
88. },
89. methods: {
90. initEditor() {
91. this.editor = new Editor({
92. el: document.getElementById(this.id),
93. ...this.editorOptions
94. })
95. if (this.value) {
96. this.editor.setValue(this.value)
97. }
98. this.editor.on('change', () => {
99. this.$emit('input', this.editor.getValue())
100. })
101. // ----------------新增↓
102. /\*
103. \* 添加自定义按钮
104. \* \*/
105. // 获取编辑器上的功能条
106. const toolbar = this.editor.getUI().getToolbar()
107. const fileDom = this.$refs.files
108. // 添加事件
109. this.editor.eventManager.addEventType('uploadEvent')
110. this.editor.eventManager.listen('uploadEvent', () => {
111. fileDom.click()
112. // Do some other thing...
113. })
114. // 添加自定义按钮 第二个参数代表位置，不传默认放在最后
115. toolbar.addButton({
116. name: 'customize',
117. className: 'tui-image',
118. event: 'uploadEvent',
119. tooltip: 'insert image',
120. // eslint-disable-next-line no-undef
121. el: '<button class="tui-image tui-toolbar-icons"></button>'
122. }, 13)
123. // 删除默认监听事件
124. this.editor.eventManager.removeEventHandler('addImageBlobHook')
125. // 添加自定义监听事件
126. this.editor.eventManager.listen('addImageBlobHook', (blob, callback) => {
127. this.upload(blob)
128. })
129. },
130. // ----------------新增↑
131. destroyEditor() {
132. if (!this.editor) return
133. this.editor.off('change')
134. this.editor.remove()
135. },
136. setValue(value) {
137. this.editor.setValue(value)
138. },
139. getValue() {
140. return this.editor.getValue()
141. },
142. setHtml(value) {
143. this.editor.setHtml(value)
144. },
145. getHtml() {
146. return this.editor.getHtml()
147. },
148. // ----------------新增↓
149. /\*
150. \* 自定义上传图片处理
151. \* \*/
152. uploadFile(e) {
153. const target = e.target
154. const file = target.files[0]
155. this.upload(file)
156. target.value = ''// 这个地方清除一下不然会有问题
157. },
158. upload(file) {
159. const formData = new FormData()
160. formData.append('file', file)
161. formData.append('dir', 'header')
162. axios({
163. method: 'post',
164. url: process.env.VUE\_APP\_UPLOAD\_URL,
165. data: formData
166. })
167. .then(res => {
168. // 上传成功地址拼接
169. const imgUrl = res.data.data
170. this.addImgToMd(imgUrl)
171. })
172. .catch(error => {
173. console.error(error.response)
174. })
175. },
176. // 添加图片到markdown
177. addImgToMd(data) {
178. const editor = this.editor.getCodeMirror()
179. const editorHtml = this.editor.getCurrentModeEditor()
180. const isMarkdownMode = this.editor.isMarkdownMode()
181. if (isMarkdownMode) {
182. editor.replaceSelection(`![img](${data})`)
183. } else {
184. const range = editorHtml.getRange()
185. const img = document.createElement('img')
186. img.src = `${data}`
187. img.alt = 'img'
188. range.insertNode(img)
189. }
190. }
191. }
192. }
193. </script>

* OSS新建一个目录，后端记得也需要配置。

# 12. 帮助中心

帮助中心无论是表结构设计，还是功能编写，都和通知公告类似。我们这里采用 tinymce 作为文本编辑器。

## 12.1 后端

### 12.1 SysHelp

java

package com.jg.pochi.pojo;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/23 20:19

\* @Version 1.0

\*/

@Data

public class SysHelp implements Serializable {

/\*\*

\* ID

\*/

private Long id;

/\*\*

\* 帮助内容

\*/

private String content;

/\*\*

\* 标题

\*/

private String title;

/\*\*

\* 创建时间

\*/

private String createdTime;

/\*\*

\* 创建人

\*/

private String createdBy;

/\*\*

\* 修改时间

\*/

private String updateTime;

/\*\*

\* 修改人

\*/

private String updateBy;

/\*\*

\* 是否删除，1是0否

\*/

private Integer deleted;

}

### 12.2 SysHelpController

java

package com.jg.pochi.controller;

import com.jg.pochi.common.Page;

import com.jg.pochi.common.Result;

import com.jg.pochi.pojo.SysHelp;

import com.jg.pochi.service.SysHelpService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/23 20:23

\* @Version 1.0

\*/

@RestController

@RequestMapping("/sysHelp")

public class SysHelpController {

@Autowired

private SysHelpService sysHelpService;

/\*\*

\* 添加帮助

\* @param sysHelp

\* @return

\*/

@RequestMapping(value = "/save", method = RequestMethod.POST)

public Result<?> save(@RequestBody SysHelp sysHelp) {

sysHelpService.save(sysHelp);

return new Result<>("添加成功");

}

/\*\*

\* 修改帮助

\* @param sysHelp

\* @return

\*/

@RequestMapping(value = "/update", method = RequestMethod.PUT)

public Result<?> update(@RequestBody SysHelp sysHelp) {

sysHelpService.update(sysHelp);

return new Result<>("修改成功");

}

/\*\*

\* 删除

\* @param id

\* @return

\*/

@RequestMapping(value = "/delete/{id}", method = RequestMethod.DELETE)

public Result<?> delete(@PathVariable Long id) {

sysHelpService.delete(id);

return new Result<>("删除成功");

}

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

public Result<SysHelp> get(@PathVariable Long id) {

SysHelp sysHelp = sysHelpService.get(id);

return new Result<>(sysHelp);

}

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

@RequestMapping(value = "/getByPage", method = RequestMethod.POST)

public Result<Page<SysHelp>> getByPage(@RequestBody Page<SysHelp> page) {

page = sysHelpService.getByPage(page);

return new Result<>(page);

}

}

### 12.3 SysHelpService

java

package com.jg.pochi.service;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysHelp;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/23 20:22

\* @Version 1.0

\*/

public interface SysHelpService {

/\*\*

\* 添加

\* @param sysHelp

\*/

void save(SysHelp sysHelp);

/\*\*

\* 修改

\* @param sysHelp

\*/

void update(SysHelp sysHelp);

/\*\*

\* 根据 id删除

\* @param id

\*/

void delete(Long id);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysHelp get(Long id);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

Page<SysHelp> getByPage(Page<SysHelp> page);

}

### 12.4 SysHelpServiceImpl

java

package com.jg.pochi.service.impl;

import com.jg.pochi.common.Page;

import com.jg.pochi.mapper.SysHelpMapper;

import com.jg.pochi.pojo.SysHelp;

import com.jg.pochi.pojo.vo.SysUserVo;

import com.jg.pochi.service.SysHelpService;

import com.jg.pochi.utils.ShiroUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/23 20:22

\* @Version 1.0

\*/

@Service

public class SysHelpServiceImpl implements SysHelpService {

@Autowired

private SysHelpMapper sysHelpMapper;

@Override

public void save(SysHelp sysHelp) {

SysUserVo loginUser = ShiroUtils.getLoginUser();

sysHelp.setCreatedBy(loginUser.getUsername());

sysHelp.setUpdateBy(loginUser.getUsername());

sysHelpMapper.save(sysHelp);

}

@Override

public void update(SysHelp sysHelp) {

SysUserVo loginUser = ShiroUtils.getLoginUser();

sysHelp.setUpdateBy(loginUser.getUsername());

sysHelpMapper.update(sysHelp);

}

@Override

public void delete(Long id) {

sysHelpMapper.delete(id);

}

@Override

public SysHelp get(Long id) {

return sysHelpMapper.get(id);

}

@Override

public Page<SysHelp> getByPage(Page<SysHelp> page) {

List<SysHelp> list = sysHelpMapper.getByPage(page);

Integer totalCount = sysHelpMapper.countByPage(page);

page.setList(list);

page.setTotalCount(totalCount);

return page;

}

}

### 12.5 SysHelpMapper

java

package com.jg.pochi.mapper;

import com.jg.pochi.common.Page;

import com.jg.pochi.pojo.SysHelp;

import org.springframework.stereotype.Component;

import java.util.List;

/\*\*

\* @Author: 杨德石

\* @Date: 2020/11/8 13:46

\* @Version 1.0

\*/

@Component

public interface SysHelpMapper {

/\*\*

\* 添加

\* @param sysHelp

\*/

void save(SysHelp sysHelp);

/\*\*

\* 修改

\* @param sysHelp

\*/

void update(SysHelp sysHelp);

/\*\*

\* 根据id删除

\* @param id

\*/

void delete(Long id);

/\*\*

\* 根据id查询

\* @param id

\* @return

\*/

SysHelp get(Long id);

/\*\*

\* 分页查询

\* @param page

\* @return

\*/

List<SysHelp> getByPage(Page<SysHelp> page);

/\*\*

\* 查询总数

\* @param page

\* @return

\*/

Integer countByPage(Page<SysHelp> page);

}

### 12.6 SysHelpMapper.xml

xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jg.pochi.mapper.SysHelpMapper">

<!-- 通用查询映射结果 -->

<resultMap id="BaseResultMap" type="com.jg.pochi.pojo.SysHelp">

<id column="id" property="id"/>

<result column="content" property="content"/>

<result column="title" property="title"/>

<result column="created\_by" property="createdBy"/>

<result column="created\_time" property="createdTime"/>

<result column="update\_time" property="updateTime"/>

<result column="update\_by" property="updateBy"/>

<result column="deleted" property="deleted"/>

</resultMap>

<insert id="save">

insert into sys\_help(content, title, created\_by, update\_by)

VALUES (#{content}, #{title}, #{createdBy}, #{updateBy})

</insert>

<update id="update">

update sys\_help

<set>

<if test="content!=null and content!=''">

content = #{content},

</if>

<if test="title!=null and title!=''">

title = #{title}

</if>

</set>

where id = #{id}

</update>

<update id="delete">

update sys\_help

set deleted = 1

where id = #{id}

</update>

<select id="get" resultMap="BaseResultMap">

select id,

content,

title,

created\_time,

created\_by,

update\_by,

update\_time

from sys\_help

where id = #{id}

</select>

<select id="getByPage" resultMap="BaseResultMap">

select id,

title,

created\_time,

created\_by,

update\_by,

update\_time

from sys\_help

where deleted = 0

<if test="params.title!=null and params.title!=null">

and title like CONCAT('%', #{params.title}, '%')

</if>

limit #{index}, #{pageSize}

</select>

<select id="countByPage" resultType="java.lang.Integer">

select count(\*)

from sys\_help

where deleted = 0

<if test="params.title!=null and params.title!=null">

and title like CONCAT('%', #{params.title}, '%')

</if>

</select>

</mapper>

## 12.2 前端

### 12.2.1 API

在 api 下创建 sysHelp.js 文件，内容如下。

js

import request from '@/utils/request'

var group\_name = 'sysHelp'

export default {

getByPage(page) { // 分页查询

return request({

url: `/${group\_name}/getByPage`,

method: 'post',

data: page

})

},

save(sysHelp) { // 保存

return request({

url: `/${group\_name}/save`,

method: 'post',

data: sysHelp

})

},

get(id) { // 根据id查询

return request({

url: `/${group\_name}/get/${id}`,

method: 'get'

})

},

update(sysHelp) { // 更新

return request({

url: `/${group\_name}/update`,

method: 'put',

data: sysHelp

})

},

deleteById(id) { // 根据id删除

return request({

url: `/${group\_name}/delete`,

method: 'put',

data: { id: id }

})

}

}

### 12.2.2 页面编写

在 views/system 下创建 help 目录，并分别创建 sys-help-list.vue、sys-help-add.vue、sys-help-update.vue、sys-help-info.vue，接着在系统中配置路由。

#### sys-help-list

vue

<template>

<div>

<!-- 搜索栏 -->

<div class="search-form">

<el-form :model="page.params" :inline="true" size="small">

<el-form-item>

<el-input v-model="page.params.title" placeholder="标题" clearable />

</el-form-item>

<el-form-item>

<el-button type="primary" icon="el-icon-search" @click="search">查询</el-button>

<el-button type="warning" icon="el-icon-refresh" @click="page.params = {}">重置</el-button>

</el-form-item>

</el-form>

</div>

<!-- 搜索栏结束 -->

<!-- 添加按钮 -->

<div class="button-group">

<el-button type="primary" size="small" icon="el-icon-plus" @click="toAdd">添加</el-button>

</div>

<!-- 添加按钮结束 -->

<!-- 数据表格开始 -->

<div class="data-table">

<el-table

header-row-class-name="pochi-table-header"

:data="dataPage.list"

stripe

style="width: 100%"

>

<el-table-column prop="title" label="标题" />

<el-table-column prop="createdBy" label="创建人" />

<el-table-column prop="createdTime" label="创建时间" />

<el-table-column prop="updateBy" label="修改人" />

<el-table-column prop="updateTime" label="修改时间" />

<el-table-column label="操作">

<template slot-scope="{row}">

<el-button type="text" icon="el-icon-document" @click="toInfo(row.id)">详情</el-button>

<el-dropdown class="handle-button">

<span class="el-dropdown-link">

操作<i class="el-icon-arrow-down el-icon--right" />

</span>

<el-dropdown-menu slot="dropdown">

<el-dropdown-item @click.native="toUpdate(row.id)">

<el-button type="text" icon="el-icon-edit">修改</el-button>

</el-dropdown-item>

<el-dropdown-item @click.native="deleteById(row.id)">

<el-button type="text" icon="el-icon-delete">删除</el-button>

</el-dropdown-item>

</el-dropdown-menu>

</el-dropdown>

</template>

</el-table-column>

</el-table>

</div>

<!-- 数据表格结束 -->

<!-- 分页组件开始 -->

<div class="pageable">

<el-pagination

:current-page="page.pageNumber"

:page-sizes="[10, 20, 30, 50]"

:page-size="10"

background

layout="total, sizes, prev, pager, next, jumper"

:total="dataPage.totalCount"

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

/>

</div>

<!-- 分页组件结束 -->

<!-- 添加弹窗 -->

<el-dialog

title="添加帮助"

:visible.sync="addDialog"

width="50%"

>

<sys-help-add @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 添加弹窗结束 -->

<!-- 修改弹窗 -->

<el-dialog

title="修改帮助"

:visible.sync="updateDialog"

width="50%"

>

<sys-help-update :active-id="activeId" @after="getByPage" @close="closeDialog" />

</el-dialog>

<!-- 修改弹窗结束 -->

<!-- 详情弹窗 -->

<el-dialog

title="帮助详情"

:visible.sync="infoDialog"

width="50%"

>

<sys-help-info :active-id="activeId" />

</el-dialog>

<!-- 详情弹窗结束 -->

</div>

</template>

<script>

import sysHelpApi from '@/api/sys-help'

import sysHelpAdd from './sys-help-add'

import sysHelpUpdate from './sys-help-update'

import sysHelpInfo from './sys-help-info'

export default {

components: {

sysHelpAdd,

sysHelpUpdate,

sysHelpInfo

},

data() {

return {

// 分页对象

page: {

// 查询参数

params: {},

// 当前页

pageNumber: 1,

// 每页条数

pageSize: 10

},

// 控制添加弹窗展示

addDialog: false,

// 控制修改弹窗展示

updateDialog: false,

// 控制详情弹窗展示

infoDialog: false,

// 数据表格分页对象

dataPage: {},

// 当前点击的帮助

activeId: null

}

},

created() {

this.getByPage()

},

methods: {

// 每页显示条数发生改变

handleSizeChange(val) {

this.page.pageSize = val

this.getByPage()

},

// 当前页发生改变

handleCurrentChange(val) {

this.page.pageNumber = val

this.getByPage()

},

// 搜索

search() {

this.page.pageNumber = 1

this.getByPage()

},

// 分页查询

getByPage() {

sysHelpApi.getByPage(this.page).then(res => {

this.dataPage = res.data

})

},

// 打开添加弹窗

toAdd() {

this.addDialog = true

},

// 打开 详情弹窗

toInfo(id) {

this.activeId = id

this.infoDialog = true

},

// 打开修改弹窗

toUpdate(id) {

this.activeId = id

this.updateDialog = true

},

// 删除

deleteById(id) {

this.$confirm('是否删除该帮助?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'error'

}).then(() => {

sysHelpApi.deleteById(id).then(res => {

this.$message.success(res.msg)

this.getByPage()

})

})

},

// 关闭弹窗

closeDialog() {

this.addDialog = false

this.updateDialog = false

this.infoDialog = false

}

}

}

</script>

<style>

</style>

#### sys-help-add

vue

<template>

<div>

<el-form :model="sysHelp" label-width="80px" size="small">

<el-form-item label="标题">

<el-input v-model="sysHelp.title" placeholder="请输入标题" clearable />

</el-form-item>

<el-form-item label="内容">

<tinymce v-model="sysHelp.content" :height="300" />

</el-form-item>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysHelpApi from '@/api/sys-help'

import Tinymce from '@/components/Tinymce'

export default {

components: { Tinymce },

data() {

return {

// 表单对象

sysHelp: {}

}

},

methods: {

// 添加

add() {

sysHelpApi.save(this.sysHelp).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

}

}

}

</script>

<style>

</style>

#### sys-help-update

vue

<template>

<div>

<el-form :model="sysHelp" label-width="80px" size="small">

<el-form-item label="标题">

<el-input v-model="sysHelp.title" placeholder="请输入标题" clearable />

</el-form-item>

<el-form-item label="内容">

<tinymce ref="content" v-model="sysHelp.content" :height="300" />

</el-form-item>

<el-form-item>

<el-button type="primary" @click="update">添加</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysHelpApi from '@/api/sys-help'

import Tinymce from '@/components/Tinymce'

export default {

components: { Tinymce },

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

// 表单对象

sysHelp: {}

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

methods: {

// 修改

update() {

sysHelpApi.update(this.sysHelp).then(res => {

this.$message.success(res.msg)

this.$emit('after')

this.$emit('close')

})

},

// 根据id查询

getById(id) {

sysHelpApi.get(id).then(res => {

this.sysHelp = res.data

this.$refs.content.setContent(this.sysHelp.content)

})

}

}

}

</script>

<style>

</style>

#### sys-help-info

vue

<template>

<div>

<el-form :model="sysHelp" label-width="80px" size="small">

<el-form-item label="标题">

{{ sysHelp.title }}

</el-form-item>

<el-form-item label="创建人">

{{ sysHelp.createdBy }}

</el-form-item>

<el-form-item label="创建时间">

{{ sysHelp.createdTime }}

</el-form-item>

<el-form-item label="修改人">

{{ sysHelp.updateBy }}

</el-form-item>

<el-form-item label="修改时间">

{{ sysHelp.updateTime }}

</el-form-item>

<el-form-item label="内容">

{{ sysHelp.content }}

</el-form-item>

</el-form>

</div>

</template>

<script>

import sysHelpApi from '@/api/sys-help'

export default {

props: {

activeId: {

type: String,

default: null

}

},

data() {

return {

// 回显对象

sysHelp: {}

}

},

watch: {

activeId: {

immediate: true,

handler: function(newVal, oldVal) {

this.getById(newVal)

}

}

},

methods: {

// 根据id查询

getById(id) {

sysHelpApi.get(id).then(res => {

this.sysHelp = res.data

})

}

}

}

</script>

<style>

</style>

### 12.2.3 Tinymce富文本

前面我们已经学习了 markdown 编辑器，现在我们来学习富文本编辑器。只需要一种编辑器的同学可以跳过本节。

#### 使用

现在市面上的富文本有很多，目前使用率最高的有三款。

* UEditor。百度出品，功能非常强大的富文本编辑器，几乎可以完全替代Word的操作。缺点是页面风格叫老 ，集成困难，一般老项目用的都是这个。
* WangEditor：国人出品的富文本编辑器，功能较少，页面风格跟layui兼容，layui项目基本都使用这个。
* Tinymce：轻量级的富文本编辑器，功能较多，页面风格与vue主流的组件库兼容，vue项目基本都使用这个。

本次课程采用 tinymce

vue-element-admin 已经为我们封装好了tinymce编辑器，直接引入使用即可。

vue

<template>

<div>

<el-form size="small" label-width="80px">

<el-row>

<el-col :span="24">

<el-form-item label="标题">

<el-input

v-model="help.title"

placeholder="请输入标题"

clearable

/>

</el-form-item>

</el-col>

</el-row>

<el-row>

<el-form-item label="内容">

<tinymce v-model="help.content" :height="500" />

</el-form-item>

</el-row>

<el-row>

<el-form-item>

<el-button type="primary" @click="add">添加</el-button>

<el-button type="warning" @click="help = {}">重置</el-button>

</el-form-item>

</el-row>

</el-form>

</div>

</template>

<script>

import Tinymce from '@/components/Tinymce'

import sysHelpApi from '@/api/sysHelp'

export default {

components: { Tinymce },

data() {

return {

help: {

}

}

},

methods: {

add() {

sysHelpApi.save(this.help).then(res => {

this.$message.success(res.msg)

this.$emit('after')

})

}

}

}

</script>

由于富文本不适合双向数据流，所以只会 watch 传入富文本的内容一次变化，之后传入内容的变化就不会再监听了，如果之后还有改变富文本内容的需求。

可以通过 this.refs.xxx.setContent() 手动来设置。

#### 图片上传

我们使用 tinymce 也需要自己定制图片上传功能。tinymce的图片上传编写难度就比 tui.editor 低得多了，开发者完全可以自己手写，而不用复制网上的代码。

图片上传的组件在 components/Tinymce/components/EditorImage.vue 里。可以看到其实就是个组件，因此我们就可以对其进行个性化的定制了。

vue

<template>

<div class="upload-container">

<el-button :style="{background:color,borderColor:color}" icon="el-icon-upload" size="mini" type="primary" @click=" dialogVisible=true">

上传图片

</el-button>

<el-dialog append-to-body :visible.sync="dialogVisible">

<el-upload

:multiple="true"

:file-list="fileList"

:show-file-list="true"

:on-remove="handleRemove"

:on-success="handleSuccess"

:before-upload="beforeUpload"

class="editor-slide-upload"

:headers="{Authorization: token}"

:data="{dir: 'tinymce'}"

:action="uploadUrl"

list-type="picture-card"

>

<el-button size="small" type="primary">

点击上传

</el-button>

</el-upload>

<el-button @click="dialogVisible = false">

取消

</el-button>

<el-button type="primary" @click="handleSubmit">

确定

</el-button>

</el-dialog>

</div>

</template>

<script>

// import { getToken } from 'api/qiniu'

import { mapGetters } from 'vuex'

export default {

name: 'EditorSlideUpload',

props: {

color: {

type: String,

default: '#1890ff'

}

},

data() {

return {

dialogVisible: false,

// 图片上传路径

uploadUrl: process.env.VUE\_APP\_UPLOAD\_URL,

listObj: {},

fileList: []

}

},

computed: {

...mapGetters([

'token'

])

},

methods: {

checkAllSuccess() {

return Object.keys(this.listObj).every(item => this.listObj[item].hasSuccess)

},

handleSubmit() {

const arr = Object.keys(this.listObj).map(v => this.listObj[v])

if (!this.checkAllSuccess()) {

this.$message('Please wait for all images to be uploaded successfully. If there is a network problem, please refresh the page and upload again!')

return

}

this.$emit('successCBK', arr)

this.listObj = {}

this.fileList = []

this.dialogVisible = false

},

handleSuccess(response, file) {

const uid = file.uid

const objKeyArr = Object.keys(this.listObj)

for (let i = 0, len = objKeyArr.length; i < len; i++) {

if (this.listObj[objKeyArr[i]].uid === uid) {

// 回显上传成功后返回的url

this.listObj[objKeyArr[i]].url = response.data

this.listObj[objKeyArr[i]].hasSuccess = true

return

}

}

},

handleRemove(file) {

const uid = file.uid

const objKeyArr = Object.keys(this.listObj)

for (let i = 0, len = objKeyArr.length; i < len; i++) {

if (this.listObj[objKeyArr[i]].uid === uid) {

delete this.listObj[objKeyArr[i]]

return

}

}

},

beforeUpload(file) {

const \_self = this

const \_URL = window.URL || window.webkitURL

const fileName = file.uid

this.listObj[fileName] = {}

return new Promise((resolve, reject) => {

const img = new Image()

img.src = \_URL.createObjectURL(file)

img.onload = function() {

\_self.listObj[fileName] = { hasSuccess: false, uid: file.uid, width: this.width, height: this.height }

}

resolve(true)

})

}

}

}

</script>

<style lang="scss" scoped>

.editor-slide-upload {

margin-bottom: 20px;

::v-deep .el-upload--picture-card {

width: 100%;

}

}

</style>