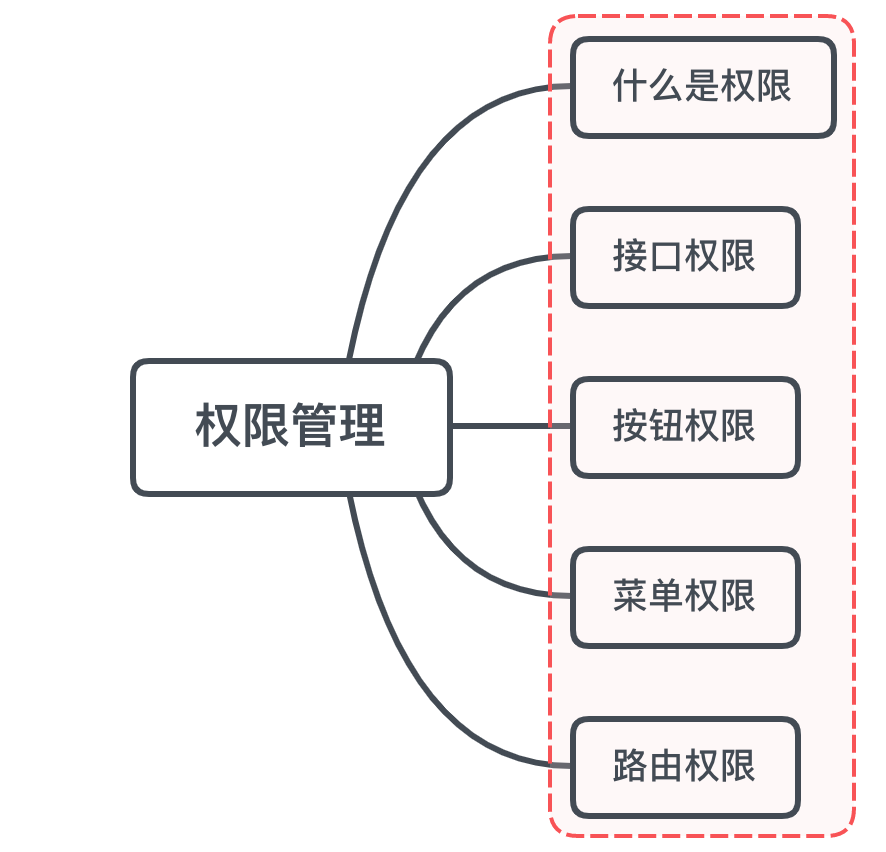
# 面试官：vue要做权限管理该怎么做？如果控制到按钮级别的权限怎么做？



## 一、是什么

权限是对特定资源的访问许可，所谓权限控制，也就是确保用户只能访问到被分配的资源

而前端权限归根结底是请求的发起权，请求的发起可能有下面两种形式触发

* 页面加载触发
* 页面上的按钮点击触发

总的来说，所有的请求发起都触发自前端路由或视图

所以我们可以从这两方面入手，对触发权限的源头进行控制，最终要实现的目标是：

* 路由方面，用户登录后只能看到自己有权访问的导航菜单，也只能访问自己有权访问的路由地址，否则将跳转 4xx 提示页
* 视图方面，用户只能看到自己有权浏览的内容和有权操作的控件
* 最后再加上请求控制作为最后一道防线，路由可能配置失误，按钮可能忘了加权限，这种时候请求控制可以用来兜底，越权请求将在前端被拦截

## 二、如何做

前端权限控制可以分为四个方面：

* 接口权限
* 按钮权限
* 菜单权限
* 路由权限

### 接口权限

接口权限目前一般采用jwt的形式来验证，没有通过的话一般返回401，跳转到登录页面重新进行登录

登录完拿到token，将token存起来，通过axios请求拦截器进行拦截，每次请求的时候头部携带token

axios.interceptors.request.use(config => {  
 config.headers['token'] = cookie.get('token')  
 return config  
})  
axios.interceptors.response.use(res=>{},{response}=>{  
 if (response.data.code === 40099 || response.data.code === 40098) { //token过期或者错误  
 router.push('/login')  
 }  
})

### 路由权限控制

**方案一**

初始化即挂载全部路由，并且在路由上标记相应的权限信息，每次路由跳转前做校验

const routerMap = [  
 {  
 path: '/permission',  
 component: Layout,  
 redirect: '/permission/index',  
 alwaysShow: true, // will always show the root menu  
 meta: {  
 title: 'permission',  
 icon: 'lock',  
 roles: ['admin', 'editor'] // you can set roles in root nav  
 },  
 children: [{  
 path: 'page',  
 component: () => import('@/views/permission/page'),  
 name: 'pagePermission',  
 meta: {  
 title: 'pagePermission',  
 roles: ['admin'] // or you can only set roles in sub nav  
 }  
 }, {  
 path: 'directive',  
 component: () => import('@/views/permission/directive'),  
 name: 'directivePermission',  
 meta: {  
 title: 'directivePermission'  
 // if do not set roles, means: this page does not require permission  
 }  
 }]  
 }]

这种方式存在以下四种缺点：

* 加载所有的路由，如果路由很多，而用户并不是所有的路由都有权限访问，对性能会有影响。
* 全局路由守卫里，每次路由跳转都要做权限判断。
* 菜单信息写死在前端，要改个显示文字或权限信息，需要重新编译
* 菜单跟路由耦合在一起，定义路由的时候还有添加菜单显示标题，图标之类的信息，而且路由不一定作为菜单显示，还要多加字段进行标识

**方案二**

初始化的时候先挂载不需要权限控制的路由，比如登录页，404等错误页。如果用户通过URL进行强制访问，则会直接进入404，相当于从源头上做了控制

登录后，获取用户的权限信息，然后筛选有权限访问的路由，在全局路由守卫里进行调用addRoutes添加路由

import router from './router'  
import store from './store'  
import { Message } from 'element-ui'  
import NProgress from 'nprogress' // progress bar  
import 'nprogress/nprogress.css'// progress bar style  
import { getToken } from '@/utils/auth' // getToken from cookie  
  
NProgress.configure({ showSpinner: false })// NProgress Configuration  
  
// permission judge function  
function hasPermission(roles, permissionRoles) {  
 if (roles.indexOf('admin') >= 0) return true // admin permission passed directly  
 if (!permissionRoles) return true  
 return roles.some(role => permissionRoles.indexOf(role) >= 0)  
}  
  
const whiteList = ['/login', '/authredirect']// no redirect whitelist  
  
router.beforeEach((to, from, next) => {  
 NProgress.start() // start progress bar  
 if (getToken()) { // determine if there has token  
 /\* has token\*/  
 if (to.path === '/login') {  
 next({ path: '/' })  
 NProgress.done() // if current page is dashboard will not trigger afterEach hook, so manually handle it  
 } else {  
 if (store.getters.roles.length === 0) { // 判断当前用户是否已拉取完user\_info信息  
 store.dispatch('GetUserInfo').then(res => { // 拉取user\_info  
 const roles = res.data.roles // note: roles must be a array! such as: ['editor','develop']  
 store.dispatch('GenerateRoutes', { roles }).then(() => { // 根据roles权限生成可访问的路由表  
 router.addRoutes(store.getters.addRouters) // 动态添加可访问路由表  
 next({ ...to, replace: true }) // hack方法 确保addRoutes已完成 ,set the replace: true so the navigation will not leave a history record  
 })  
 }).catch((err) => {  
 store.dispatch('FedLogOut').then(() => {  
 Message.error(err || 'Verification failed, please login again')  
 next({ path: '/' })  
 })  
 })  
 } else {  
 // 没有动态改变权限的需求可直接next() 删除下方权限判断 ↓  
 if (hasPermission(store.getters.roles, to.meta.roles)) {  
 next()//  
 } else {  
 next({ path: '/401', replace: true, query: { noGoBack: true }})  
 }  
 // 可删 ↑  
 }  
 }  
 } else {  
 /\* has no token\*/  
 if (whiteList.indexOf(to.path) !== -1) { // 在免登录白名单，直接进入  
 next()  
 } else {  
 next('/login') // 否则全部重定向到登录页  
 NProgress.done() // if current page is login will not trigger afterEach hook, so manually handle it  
 }  
 }  
})  
  
router.afterEach(() => {  
 NProgress.done() // finish progress bar  
})

按需挂载，路由就需要知道用户的路由权限，也就是在用户登录进来的时候就要知道当前用户拥有哪些路由权限

这种方式也存在了以下的缺点：

* 全局路由守卫里，每次路由跳转都要做判断
* 菜单信息写死在前端，要改个显示文字或权限信息，需要重新编译
* 菜单跟路由耦合在一起，定义路由的时候还有添加菜单显示标题，图标之类的信息，而且路由不一定作为菜单显示，还要多加字段进行标识

### 菜单权限

菜单权限可以理解成将页面与理由进行解耦

#### 方案一

菜单与路由分离，菜单由后端返回

前端定义路由信息

{  
 name: "login",  
 path: "/login",  
 component: () => import("@/pages/Login.vue")  
}

name字段都不为空，需要根据此字段与后端返回菜单做关联，后端返回的菜单信息中必须要有name对应的字段，并且做唯一性校验

全局路由守卫里做判断

function hasPermission(router, accessMenu) {  
 if (whiteList.indexOf(router.path) !== -1) {  
 return true;  
 }  
 let menu = Util.getMenuByName(router.name, accessMenu);  
 if (menu.name) {  
 return true;  
 }  
 return false;  
  
}  
  
Router.beforeEach(async (to, from, next) => {  
 if (getToken()) {  
 let userInfo = store.state.user.userInfo;  
 if (!userInfo.name) {  
 try {  
 await store.dispatch("GetUserInfo")  
 await store.dispatch('updateAccessMenu')  
 if (to.path === '/login') {  
 next({ name: 'home\_index' })  
 } else {  
 //Util.toDefaultPage([...routers], to.name, router, next);  
 next({ ...to, replace: true })//菜单权限更新完成,重新进一次当前路由  
 }  
 }   
 catch (e) {  
 if (whiteList.indexOf(to.path) !== -1) { // 在免登录白名单，直接进入  
 next()  
 } else {  
 next('/login')  
 }  
 }  
 } else {  
 if (to.path === '/login') {  
 next({ name: 'home\_index' })  
 } else {  
 if (hasPermission(to, store.getters.accessMenu)) {  
 Util.toDefaultPage(store.getters.accessMenu,to, routes, next);  
 } else {  
 next({ path: '/403',replace:true })  
 }  
 }  
 }  
 } else {  
 if (whiteList.indexOf(to.path) !== -1) { // 在免登录白名单，直接进入  
 next()  
 } else {  
 next('/login')  
 }  
 }  
 let menu = Util.getMenuByName(to.name, store.getters.accessMenu);  
 Util.title(menu.title);  
});  
  
Router.afterEach((to) => {  
 window.scrollTo(0, 0);  
});

每次路由跳转的时候都要判断权限，这里的判断也很简单，因为菜单的name与路由的name是一一对应的，而后端返回的菜单就已经是经过权限过滤的

如果根据路由name找不到对应的菜单，就表示用户有没权限访问

如果路由很多，可以在应用初始化的时候，只挂载不需要权限控制的路由。取得后端返回的菜单后，根据菜单与路由的对应关系，筛选出可访问的路由，通过addRoutes动态挂载

这种方式的缺点：

* 菜单需要与路由做一一对应，前端添加了新功能，需要通过菜单管理功能添加新的菜单，如果菜单配置的不对会导致应用不能正常使用
* 全局路由守卫里，每次路由跳转都要做判断

#### 方案二

菜单和路由都由后端返回

前端统一定义路由组件

const Home = () => import("../pages/Home.vue");  
const UserInfo = () => import("../pages/UserInfo.vue");  
export default {  
 home: Home,  
 userInfo: UserInfo  
};

后端路由组件返回以下格式

[  
 {  
 name: "home",  
 path: "/",  
 component: "home"  
 },  
 {  
 name: "home",  
 path: "/userinfo",  
 component: "userInfo"  
 }  
]

在将后端返回路由通过addRoutes动态挂载之间，需要将数据处理一下，将component字段换为真正的组件

如果有嵌套路由，后端功能设计的时候，要注意添加相应的字段，前端拿到数据也要做相应的处理

这种方法也会存在缺点：

* 全局路由守卫里，每次路由跳转都要做判断
* 前后端的配合要求更高

### 按钮权限

#### 方案一

按钮权限也可以用v-if判断

但是如果页面过多，每个页面页面都要获取用户权限role和路由表里的meta.btnPermissions，然后再做判断

这种方式就不展开举例了

#### 方案二

通过自定义指令进行按钮权限的判断

首先配置路由

{  
 path: '/permission',  
 component: Layout,  
 name: '权限测试',  
 meta: {  
 btnPermissions: ['admin', 'supper', 'normal']  
 },  
 //页面需要的权限  
 children: [{  
 path: 'supper',  
 component: \_import('system/supper'),  
 name: '权限测试页',  
 meta: {  
 btnPermissions: ['admin', 'supper']  
 } //页面需要的权限  
 },  
 {  
 path: 'normal',  
 component: \_import('system/normal'),  
 name: '权限测试页',  
 meta: {  
 btnPermissions: ['admin']  
 } //页面需要的权限  
 }]  
}

自定义权限鉴定指令

import Vue from 'vue'  
/\*\*权限指令\*\*/  
const has = Vue.directive('has', {  
 bind: function (el, binding, vnode) {  
 // 获取页面按钮权限  
 let btnPermissionsArr = [];  
 if(binding.value){  
 // 如果指令传值，获取指令参数，根据指令参数和当前登录人按钮权限做比较。  
 btnPermissionsArr = Array.of(binding.value);  
 }else{  
 // 否则获取路由中的参数，根据路由的btnPermissionsArr和当前登录人按钮权限做比较。  
 btnPermissionsArr = vnode.context.$route.meta.btnPermissions;  
 }  
 if (!Vue.prototype.$\_has(btnPermissionsArr)) {  
 el.parentNode.removeChild(el);  
 }  
 }  
});  
// 权限检查方法  
Vue.prototype.$\_has = function (value) {  
 let isExist = false;  
 // 获取用户按钮权限  
 let btnPermissionsStr = sessionStorage.getItem("btnPermissions");  
 if (btnPermissionsStr == undefined || btnPermissionsStr == null) {  
 return false;  
 }  
 if (value.indexOf(btnPermissionsStr) > -1) {  
 isExist = true;  
 }  
 return isExist;  
};  
export {has}

在使用的按钮中只需要引用v-has指令

<el-button @click='editClick' type="primary" v-has>编辑</el-button>

### 小结

关于权限如何选择哪种合适的方案，可以根据自己项目的方案项目，如考虑路由与菜单是否分离

权限需要前后端结合，前端尽可能的去控制，更多的需要后台判断

## 参考文献

* https://mp.weixin.qq.com/s/b-D2eH1mLwL\_FkaZwjueSw
* https://segmentfault.com/a/1190000020887109
* https://juejin.cn/post/6844903648057622536#heading-6