

# 导航菜单功能

## 后台权限

参考 <https://www.django-rest-framework.org/api-guide/permissions/>

```
1 REST_FRAMEWORK = {
2     'EXCEPTION_HANDLER': 'utils.exceptions.global_exception_handler',
3     'DEFAULT_AUTHENTICATION_CLASSES': [
4         'rest_framework_simplejwt.authentication.JWTAuthentication'
5     ],
6     'DEFAULT_PERMISSION_CLASSES': [
7         'rest_framework.permissions.IsAuthenticated', # 全局，要求被认证，也就是登
            录成功
8     ]
9 }
```

## 主路由

```
1 from django.contrib import admin
2 from django.urls import path, include
3 from rest_framework_simplejwt.views import (
4     TokenObtainPairView,
5     TokenRefreshView,
6 )
7 from user.views import test
8
9 tobview = TokenObtainPairView.as_view() # 这个视图函数生成一次就可以了，可以调用n次
10
11 urlpatterns = [
12     # path('admin/', admin.site.urls),
13     path('login/', tobview, name='login'),
14     path('token/', tobview, name='token_obtain_pair'), # 获取
15     path('token/refresh/', TokenRefreshView.as_view(),
16         name='token_refresh'), # 刷新
17     path('test', test),
18     path('users/', include('user.urls')) # 二级路由到users/
19 ]
```

## 需求

只要是认证的用户都可以看到菜单项，管理员无视权限能看到所有项，普通用户能看到部分项（以后可以按照用户、组分配）。

后台构建一个视图类，菜单项功能的权限要求就是 被认证的、 管理员

user二级路由

```

1 from django.urls import path
2 from .views import menulist_view
3
4 urlpatterns = [
5     path('menulist/', menulist_view),
6 ]

```

```

1 from rest_framework.decorators import api_view, permission_classes
2 from rest_framework.response import Response
3 from rest_framework.request import Request
4 from rest_framework.permissions import IsAuthenticated, IsAdminUser
5
6 @api_view(['GET'])
7 @permission_classes([IsAuthenticated, IsAdminUser]) # 覆盖全局配置
8 def menulist_view(request: Request):
9     user = request.user
10    auth = request.auth
11    print(user, auth)
12    print(dir(user)) # .is_superuser
13
14    return Response({'test': 300})

```

没有token访问会抛出 `rest_framework.exceptions.NotAuthenticated`，为异常模块定义一个新的映射

utils/exceptions.py

```

1 class NotAuthenticated(MagBaseException):
2     code = 3
3     message = '未登录，请重新登录'
4
5     # 内部异常暴露细节
6     exc_map = {
7         'AuthenticationFailed': InvalidUsernameOrPassword,
8         'InvalidToken': InvalidToken, # token过期
9         'NotAuthenticated': NotAuthenticated, # 无token登录
10    }

```

## 前台菜单项

### 数据设计

```

<el-menu background-color="#123" text-color="#fff" acti
  <el-submenu index="1">
    <template slot="title">
      <i class="el-icon-menu"></i>
      <span slot="title">用户管理</span>
    </template>
    <el-menu-item index="1-1">用户列表</el-menu-item>
    <el-menu-item index="1-2">用户列表</el-menu-item>
  </el-submenu>
  <el-submenu index="2">
    <template slot="title">
      <i class="el-icon-menu"></i>
      <span slot="title">导航2</span>
    </template>
    <el-menu-item index="2-1">选项1</el-menu-item>
    <el-menu-item index="2-2">选项2</el-menu-item>
  </el-submenu>
</el-menu>

```

从上图可以看出要为菜单项构造一个层级结构，简化设计，2层就可以了。

菜单项有层级，每一项可以有id（对应index）、itemName项名、path跳转路径（可以为null）、children子项列表。例如

```

1  {
2    "data": [
3      {
4        "id": 101,
5        "itemName": "用户管理",
6        "path": null,
7        "children": [
8          {
9            "id": 102,
10           "itemName": "权限管理",
11           "path": null,
12           "children": []
13         }
14       ]
15     },
16     {
17       "id": 201,
18       "itemName": "仪表盘",
19       "path": null,
20       "children": []
21     }
22   ]
23 }

```

## 后端菜单生成

由于每一个菜单项都是一个字典，所以菜单项类直接就是dict的子类

```
1 from rest_framework.decorators import api_view, permission_classes
2 from rest_framework.response import Response
3 from rest_framework.request import Request
4 from rest_framework.permissions import IsAuthenticated, IsAdminUser
5
6 class PermissionItem(dict):
7     def __init__(self, id, itemName, path=None):
8         super().__init__()
9         self.id = id
10        self.itemName = itemName
11        self.path = path
12        self.children = []
13
14    def __setattr__(self, key, value):
15        """将属性变成kv对"""
16        self[key] = value
17
18    def __getattr__(self, key):
19        return self[key]
20
21    def append(self, child):
22        self.children.append(child)
23
24 @api_view(['GET'])
25 @permission_classes([IsAuthenticated, IsAdminUser]) # 覆盖全局配置
26 def menulist_view(request: Request):
27     user = request.user
28     auth = request.auth
29     print(user, auth)
30     print(dir(user)) # .is_superuser
31     menulist = {
32         'data': []
33     }
34     if user.is_superuser: # 用户管理必须管理员
35         p1 = PermissionItem(101, '用户管理')
36         p1.append(PermissionItem(102, '用户列表', '/users/'))
37         p1.append(PermissionItem(103, '角色列表', '/users/roles/'))
38         p1.append(PermissionItem(104, '权限列表', '/users/perms/'))
39         menulist['data'].append(p1)
40
41     return Response(menulist)
```

GET http://127.0.0.1:8000/users/menulist/

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Accept Accept-Encoding ① Connection ① Authorization

Key Value Description

Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2t1b90e...

Body Cookies Headers (9) Test Results Status: 200 OK Time: 15 ms Size: 555 B S

Pretty Raw Preview Visualize JSON

```

2  "data": [
3    {
4      "id": 101,
5      "itemName": "用户管理",
6      "path": null,
7      "children": [
8        {
9          "id": 102,
10         "itemName": "用户列表",
11         "path": "/users/",

```

## 前台解析渲染

返回的数据是列表，从中直接提取数据循环即可

v-for <https://cn.vuejs.org/v2/api/#v-for>

```

1  <el-menu background-color="#123" text-color="#fff" active-text-
   color="#ffd04b">
2    <el-submenu v-for="item in menulist" :index="item.id + ''" :key="item.id">
3      <template slot="title"><i class="el-icon-menu"></i><span slot="title">{{
   item.itemName }}</span></template>
4      <el-menu-item v-for="subItem in item.children" :index="subItem.id + ''"
   :key="subItem.id">
5        <template slot="title"><i class="el-icon-menu"></i><span slot="title">
   {{ subItem.itemName }}</span></template>
6        </el-menu-item>
7      </el-submenu>
8    </el-menu>

```

v-for迭代是需要v-bind:key，key唯一就行，不需要是字符串，但index是el-submenu属性，要求是字符串。

<template slot="title"><i class="el-icon-menu"></i><span slot="title">{{ item.itemName }}</span></template>，结构是template里面要用span把标题包起来。

Home.vue

```

1  <template>
2    <el-container>
3      <el-header>
4        <div class="logo">
5          
6          <div class="title">马哥教育猛犸运维系统管理平台</div>
7        </div>
8        <div class="info">

```

```

9       <el-button type="info" @click="logout">退出</el-button>
10     </div>
11   </el-header>
12   <el-container>
13     <el-aside width="320px">
14       <el-menu default-active="2-2" background-color="#123" text-
15       color="#fff" active-text-color="#ffd04b">
16         <el-submenu v-for="item in menulist" :index="item.id + ''"
17         :key="item.id">
18           <template slot="title">
19             <i class="el-icon-menu"></i><span slot="title">{{
20             item.itemName }}</span>
21           </template>
22           <el-menu-item v-for="subItem in item.children"
23           :index="subItem.id + ''" :key="subItem.id">
24             <template slot="title">
25               <i class="el-icon-menu"></i><span slot="title">{{
26               subItem.itemName }}</span>
27             </template>
28           </el-menu-item>
29         </el-submenu>
30       </el-menu>
31     </el-aside>
32     <el-main>Main</el-main>
33   </el-container>
34 </el-container>
35 </template>
36
37 <script>
38 export default {
39   created() {
40     this.getMenulist() // vue组件创建时查菜单数据
41   },
42   data() {
43     return {
44       menulist: [] // 菜单数据
45     }
46   },
47   methods: {
48     async getMenulist() {
49       const { data: response } = await this.$http.get('users/menulist/')
50       if (response.code) {
51         return this.$message.error(response.message)
52       }
53       this.menulist = response.data // 菜单项数组
54     },
55     logout() {
56       window.localStorage.removeItem('token')
57       this.$router.push('/login')
58     }
59   }
60 }
61 </script>

```

## 折叠菜单

参考 <https://element.eleme.cn/#/zh-CN/component/menu#zhe-die>

为Menu组件增加一个绑定属性 :collapse="isCollapse" 才能折叠, 是否开启动画 :collapse-transition="true"

Home.vue

```
1 <template>
2   <el-container>
3     <el-header>
4       <div class="logo">
5         
6         <div class="title">马哥教育猛犸运维系统管理平台</div>
7         <div><i :class="isCollapsed ? 'el-icon-s-unfold' : 'el-icon-s-
fold'" @click="toggleMenu"></i></div>
8       </div>
9       <div class="info">
10        <el-button type="info" @click="logout">退出</el-button>
11      </div>
12    </el-header>
13    <el-container>
14      <el-aside :width="isCollapsed ? '64px' : '200px'">
15        <el-menu
16          default-active="101"
17          background-color="#123"
18          text-color="#fff"
19          active-text-color="#ffd04b"
20          :collapse="isCollapsed"
21          :collapse-transition="false"
22        >
23          <el-submenu v-for="item in menulist" :index="item.id + ''"
:key="item.id">
24            <template slot="title">
25              <i class="el-icon-menu"></i><span slot="title">{{
item.itemName }}</span>
26            </template>
27            <el-menu-item v-for="subItem in item.children"
:index="subItem.id + ''" :key="subItem.id">
28              <template slot="title">
29                <i class="el-icon-menu"></i><span slot="title">{{
subItem.itemName }}</span>
30              </template>
31            </el-menu-item>
32          </el-submenu>
33        </el-menu>
34      </el-aside>
35      <el-main>Main</el-main>
36    </el-container>
37  </el-container>
38 </template>
39
40 <script>
41 export default {
42   created() {
43     // vue组件创建时查菜单数据
```

```

44     this.getMenuList()
45   },
46   data() {
47     return {
48       menulist: [], // 菜单数据
49       isCollapsed: true
50     }
51   },
52   methods: {
53     async getMenuList() {
54       const { data: response } = await this.$http.get('users/menulist/')
55       if (response.code) {
56         return this.$message.error(response.message)
57       }
58       this.menulist = response.data // 菜单项数组
59     },
60     logout() {
61       window.localStorage.removeItem('token')
62       this.$router.push('/login')
63     },
64     toggleMenu() {
65       this.isCollapsed = !this.isCollapsed
66     }
67   }
68 }
69 </script>
70
71 <style lang="less" scoped>
72 .el-container {
73   height: 100%;
74 }
75
76 .el-header {
77   display: flex;
78   justify-content: space-between; // 主轴两端对齐
79   align-items: center; // 侧轴居中
80   padding-left: 5px;
81   .logo {
82     display: flex;
83     img {
84       width: 30px;
85     }
86     .title {
87       font-size: 24px;
88       margin-left: 5px;
89     }
90     .el-icon-s-unfold,
91     .el-icon-s-fold {
92       font-size: 32px;
93       margin-left: 10px;
94     }
95   }
96 }
97
98 .el-aside {
99   background-color: #123;
100 }
101

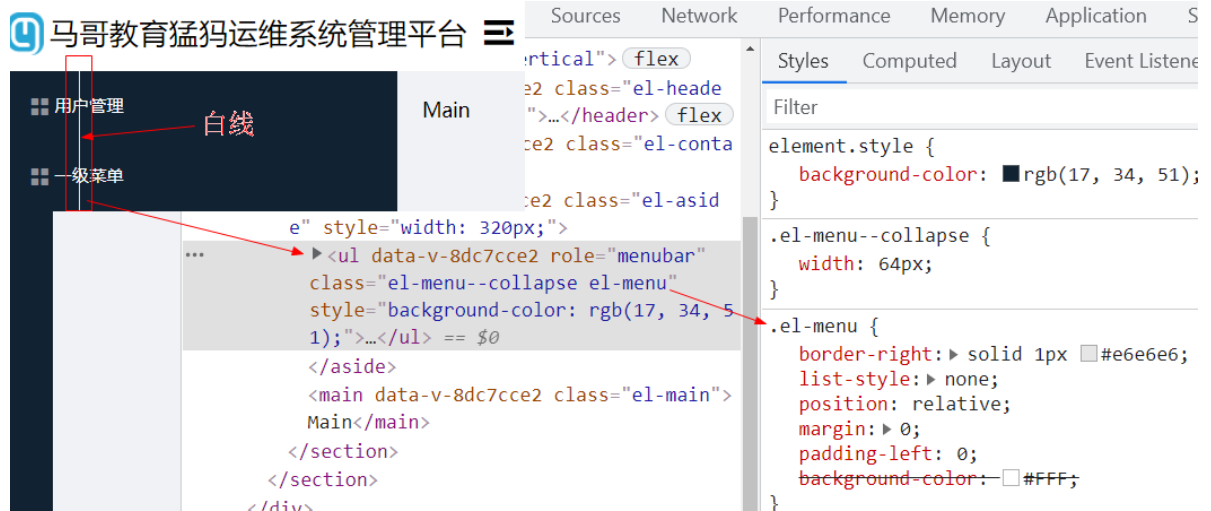
```



```

102 .el-main {
103   background-color: #f0f2f5;
104 }
105
106 .el-menu {
107   border-right: none; // 消除菜单右侧白线
108 }
109 </style>

```



## 嵌套路由

### 欢迎页

新建src/components/Welcome.vue

```

1 <template>
2   <div>
3     <h3>欢迎访问马哥教育猛犸运维平台系统 by wayne</h3>
4   </div>
5 </template>
6
7 <script>
8   export default {
9
10  }
11 </script>
12
13 <style>
14
15 </style>

```

## 嵌套路由

子路由参考 <https://router.vuejs.org/zh/guide/essentials/nested-routes.html>

1	/user/foo/profile		/user/foo/posts
2	+-----+		+-----+
3	User		User
4	+-----+		+-----+
5	Profile	+----->	Posts
6			
7	+-----+		+-----+
8	+-----+		+-----+

也就是说，前端路径发生了变化，User组件不变，其中的子组件发生了切换。

src/router/index.js

```

1  import Vue from 'vue'
2  import VueRouter from 'vue-router'
3  import Login from '../components/Login.vue'
4  import Home from '../components/Home.vue'
5  import Welcome from '../components/welcome.vue'
6
7  Vue.use(VueRouter)
8
9  const routes = [
10   { path: '/', redirect: '/login' },
11   { path: '/login', component: Login },
12   {
13     path: '/home',
14     component: Home,
15     redirect: '/welcome', // 访问/home重定向到/welcome，进入子路由
16     children: [
17       { path: '/welcome', component: Welcome }
18     ]
19   }
20 ]
21
22 const router = new VueRouter({
23   routes
24 })
25
26 // 挂载全局导航守卫
27 router.beforeEach((to, from, next) => { // from从哪里来，to去哪里，next函数跳转
28   if (to.path === '/login') {
29     next()
30   }
31   // 读取token
32   const token = window.localStorage.getItem('token')
33   if (!token) {
34     next('/login')
35   }
36   next()
37 })
38
39 export default router

```

Home.vue中

- 在el-main中插入 `<router-view></router-view>`，作为子路由显示的地方
- 在el-menu中 `:router="true"`，开启路由
- 一旦开启路由后，点击el-menu-item菜单项就会按照index跳转。所以，把el-menu-item的index改为 `<el-menu-item v-for="subItem in item.children" :index="subItem.path" :key="subItem.id">`

```

1  <template>
2    <el-container>
3      <el-header>
4        <div class="logo">
5          
6          <div class="title">马哥教育猛犸运维系统管理平台</div>
7          <div><i :class="isCollapsed ? 'el-icon-s-unfold' : 'el-icon-s-fold'"
@click="toggleMenu"></i></div>
8        </div>
9        <div class="info">
10         <el-button type="info" @click="logout">退出</el-button>
11       </div>
12     </el-header>
13     <el-container>
14       <el-aside :width="isCollapsed ? '64px' : '200px'">
15         <el-menu
16           default-active="101"
17           background-color="#123"
18           text-color="#fff"
19           active-text-color="#ffd04b"
20           :collapse="isCollapsed"
21           :collapse-transition="false"
22           :router="true"
23         >
24           <el-submenu v-for="item in menulist" :index="item.id + '"
:key="item.id">
25             <template slot="title">
26               <i class="el-icon-menu"></i><span slot="title">{{
item.itemName }}</span>
27             </template>
28             <el-menu-item v-for="subItem in item.children"
:index="subItem.path" :key="subItem.id">
29               <template slot="title">
30                 <i class="el-icon-menu"></i><span slot="title">{{
subItem.itemName }}</span>
31               </template>
32             </el-menu-item>
33           </el-submenu>
34         </el-menu>
35       </el-aside>
36       <el-main>
37         <!-- 嵌套路由 -->
38         <router-view></router-view>
39       </el-main>
40     </el-container>
41   </el-container>
42 </template>
43
44 <script>
45 export default {
46   created() {

```

```

47 // vue组件创建时查菜单数据
48 this.getMenuList()
49 },
50 data() {
51   return {
52     menulist: [], // 菜单数据
53     isCollapsed: false
54   }
55 },
56 methods: {
57   async getMenuList() {
58     const { data: response } = await this.$http.get('users/menulist/')
59     if (response.code) {
60       return this.$message.error(response.message)
61     }
62     this.menulist = response.data // 菜单项数组
63   },
64   logout() {
65     window.localStorage.removeItem('token')
66     this.$router.push('/login')
67   },
68   toggleMenu() {
69     this.isCollapsed = !this.isCollapsed
70   }
71 }
72 }
73 </script>

```

## 失败重登录

访问后台资源的时候，验证了token，返回关于token的异常json，其中包含code、message。

使用axios全局响应拦截器处理。后台对异常的code做一下规定，小于100是与登录有关的异常，前端就会重登录。

后台utils/exceptions.py

```

1 from django.http import Http404
2 from django.core.exceptions import PermissionDenied
3 from rest_framework.views import set_rollback
4 from rest_framework import exceptions
5 from rest_framework.views import Response
6
7 class MagBaseException(exceptions.APIException):
8     """基类定义基本的异常"""
9     code = 10000 # code为0表示正常，非0都是错误
10    message = '非法请求' # 错误描述
11
12    @classmethod
13    def get_message(cls):
14        return {'code': cls.code, 'message': cls.message}
15
16 class InvalidUsernameOrPassword(MagBaseException):
17     code = 1
18     message = '用户名或密码错误'

```

```

19
20 class InvalidToken(MagBaseException):
21     code = 2
22     message = '认证无效, 请重新登录'
23
24 class NotAuthenticated(MagBaseException):
25     code = 3
26     message = '未登录, 请重新登录'
27
28 # code < 100, 前端清除token, 跳转到登录页
29
30 # 内部异常暴露细节
31 exc_map = {
32     'AuthenticationFailed': InvalidUsernameOrPassword,
33     'InvalidToken': InvalidToken,          # token过期
34     'NotAuthenticated': NotAuthenticated, # 无token登录
35 }
36
37
38 def global_exception_handler(exc, context):
39     """
40     全局异常处理
41
42     照抄rest_framework.views.exception_handler, 略作修改
43     不管什么异常这里统一处理。根据不同类型显示不同的
44     为了前端解析方便, 这里响应的状态码采用默认的200
45     异常对应处理后返回对应的错误码和错误描述
46     异常找不到对应就返回缺省
47     """
48     if isinstance(exc, Http404):
49         exc = exceptions.NotFound()
50     elif isinstance(exc, PermissionDenied):
51         exc = exceptions.PermissionDenied()
52
53     print('异常', '=' * 30)
54     print(type(exc), exc.__class__.__name__, exc.__dict__)
55     print('=' * 30)
56
57     if isinstance(exc, exceptions.APIException):
58         headers = {}
59         if getattr(exc, 'auth_header', None):
60             headers['WWW-Authenticate'] = exc.auth_header
61         if getattr(exc, 'wait', None):
62             headers['Retry-After'] = '%d' % exc.wait
63
64         if isinstance(exc.detail, (list, dict)):
65             data = exc.detail
66         else:
67             data = {'detail': exc.detail}
68
69         set_rollback()
70
71         if isinstance(exc, MagBaseException):
72             errmsg = exc.get_message()
73         else:
74             errmsg = exc_map.get(exc.__class__.__name__,
MagBaseException).get_message()
75         return Response(errmsg, status=200) # 状态恒为200

```

```
76         #return Response(data, status=exc.status_code, headers=headers)
77
78     return None
```

前台src/main.js

```
1  import Vue from 'vue'
2  import App from './App.vue'
3  import router from './router'
4  import './plugins/element.js'
5  import './assets/css/main.css'
6  import axios from 'axios'
7
8  // 请求拦截器，只要发起异步请求都要带上这个token
9  axios.interceptors.request.use(function(config) {
10    // 在发送请求之前做些什么
11    config.headers.Authorization = 'Bearer ' +
    window.localStorage.getItem('token')
12    return config
13  })
14  // 响应拦截器，只要有错误码小于100都和登录有关，要求重新登录
15  axios.interceptors.response.use(function(response) {
16    // 对响应数据做点什么
17    if (response.data.code && response.data.code < 100) {
18      router.push('/login')
19    } else {
20      return response
21    }
22  })
23  // baseUrl指向后台服务
24  axios.defaults.baseURL = '/api/v1/'
25  // 为Vue类增加全局属性$http，这样所有组件实例都可以使用该属性了
26  Vue.prototype.$http = axios
27
28  Vue.config.productionTip = false
29
30  new Vue({
31    router,
32    render: h => h(App)
33  }).$mount('#app')
```

到此，主页开发完成，分别提交前后端代码。