导航菜单功能

后台权限

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

```
REST_FRAMEWORK = {
    'EXCEPTION_HANDLER': 'utils.exceptions.global_exception_handler',
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework_simplejwt.authentication.JWTAuthentication'
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated', # 全局，要求被认证，也就是登录成功
    ]
}
```

主路由

```
from django.contrib import admin
from django.urls import path, include
from rest_framework_simplejwt.views import (
    TokenObtainPairView,
    TokenRefreshView,
)
from user.views import test

tobview = TokenObtainPairView.as_view() # 这个视图函数生成一次就可以了，可以调用n次

urlpatterns = [
    # path('admin/', admin.site.urls),
    path('login/', tobview, name='login'),
    path('token/', tobview, name='token_obtain_pair'),  # 获取
    path('token/refresh/', TokenRefreshView.as_view(), name='token_refresh'), # 刷新
    path('test', test),
    path('users/', include('user.urls')) # 二级路由到users/
]
```

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

```python
from django.urls import path
from .views import menulist_view

urlpatterns = [
    path('menulist/', menulist_view),
]
```

```python
from rest_framework.decorators import api_view, permission_classes
from rest_framework.response import Response
from rest_framework.request import Request
from rest_framework.permissions import IsAuthenticated, IsAdminUser

@api_view(['GET'])
@permission_classes([IsAuthenticated, IsAdminUser]) # 覆盖全局配置
def menulist_view(request:Request):
    user = request.user
    auth = request.auth
    print(user, auth)
    print(dir(user)) # .is_superuser

    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

```python
class NotAuthenticated(MagBaseException):
    code = 3
    message = '未登录，请重新登录'

# 内部异常暴露细节
exc_map = {
    'AuthenticationFailed': InvalidUsernameOrPassword,
    'InvalidToken': InvalidToken,          # token过期
    'NotAuthenticated': NotAuthenticated, # 无token登录
}
```

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。

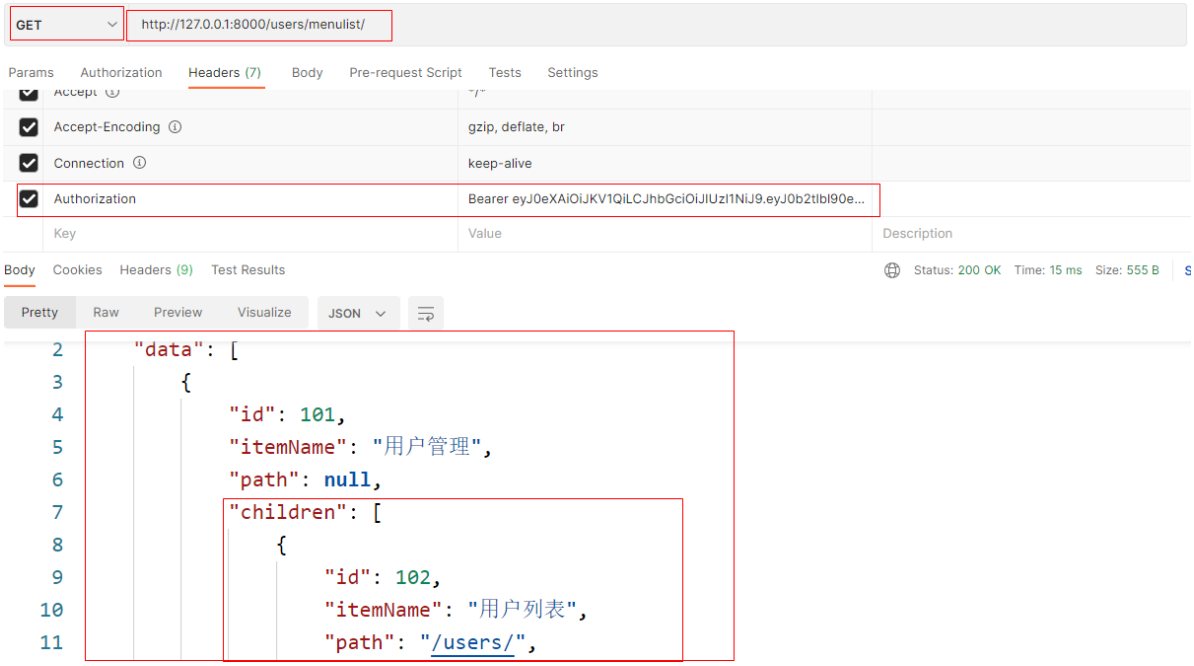
菜单项有层级，每一项可以有id（对应index）、itemName项名、path跳转路径（可以为null）、children子项列表。例如

```
{
    "data": [
        {
            "id": 101,
            "itemName": "用户管理",
            "path": null,
            "children": [
                {
                    "id": 102,
                    "itemName": "权限管理",
                    "path": null,
                    "children": []
                }
            ]
        },
        {
            "id": 201,
            "itemName": "仪表盘",
            "path": null,
            "children": []
        }
    ]
}
```

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

```python
from rest_framework.decorators import api_view, permission_classes
from rest_framework.response import Response
from rest_framework.request import Request
from rest_framework.permissions import IsAuthenticated, IsAdminUser

class PermissionItem(dict):
    def __init__(self, id, itemName, path=None):
        super().__init__()
        self.id = id
        self.itemName = itemName
        self.path = path
        self.children = []

    def __setattr__(self, key, value):
        """将属性变成kv对"""
        self[key] = value

    def __getattr__(self, key):
        return self[key]

    def append(self, child):
        self.children.append(child)

@api_view(['GET'])
@permission_classes([IsAuthenticated, IsAdminUser]) # 覆盖全局配置
def menulist_view(request:Request):
    user = request.user
    auth = request.auth
    print(user, auth)
    print(dir(user)) # .is_superuser
    menulist = {
        'data':[]
    }
    if user.is_superuser: # 用户管理必须管理员
        p1 = PermissionItem(101, '用户管理')
        p1.append(PermissionItem(102, '用户列表', '/users/'))
        p1.append(PermissionItem(103, '角色列表', '/users/roles/'))
        p1.append(PermissionItem(104, '权限列表', '/users/perms/'))
        menulist['data'].append(p1)

    return Response(menulist)
```



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

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

```
<template>
  <el-container>
    <el-header>
      <div class="logo">
        <img src="../assets/logo.png" alt="logo" />
        <div class="title">马哥教育猛犸运维系统管理平台</div>
      </div>
      <div class="info">
```

```
        <el-button type="info" @click="logout">退出</el-button>
      </div>
    </el-header>
    <el-container>
      <el-aside width="320px">
        <el-menu default-active="2-2" background-color="#123" text-color="#fff" active-text-color="#ffd04b">
          <el-submenu v-for="item in menulist" :index="item.id + ''" :key="item.id">
            <template slot="title">
              <i class="el-icon-menu"></i><span slot="title">{{ item.itemName }}</span>
            </template>
            <el-menu-item v-for="subItem in item.children" :index="subItem.id + ''" :key="subItem.id">
              <template slot="title">
                <i class="el-icon-menu"></i><span slot="title">{{ subItem.itemName }}</span>
              </template>
            </el-menu-item>
          </el-submenu>
        </el-menu>
      </el-aside>
      <el-main>Main</el-main>
    </el-container>
  </el-container>
</template>

<script>
export default {
  created() {
    this.getMenulist() // vue组件创建时查菜单数据
  },
  data() {
    return {
      menulist: [] // 菜单数据
    }
  },
  methods: {
    async getMenulist() {
      const { data: response } = await this.$http.get('users/menulist/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.menulist = response.data // 菜单项数组
    },
    logout() {
      window.localStorage.removeItem('token')
      this.$router.push('/login')
    }
  }
}
</script>
```

折叠菜单

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

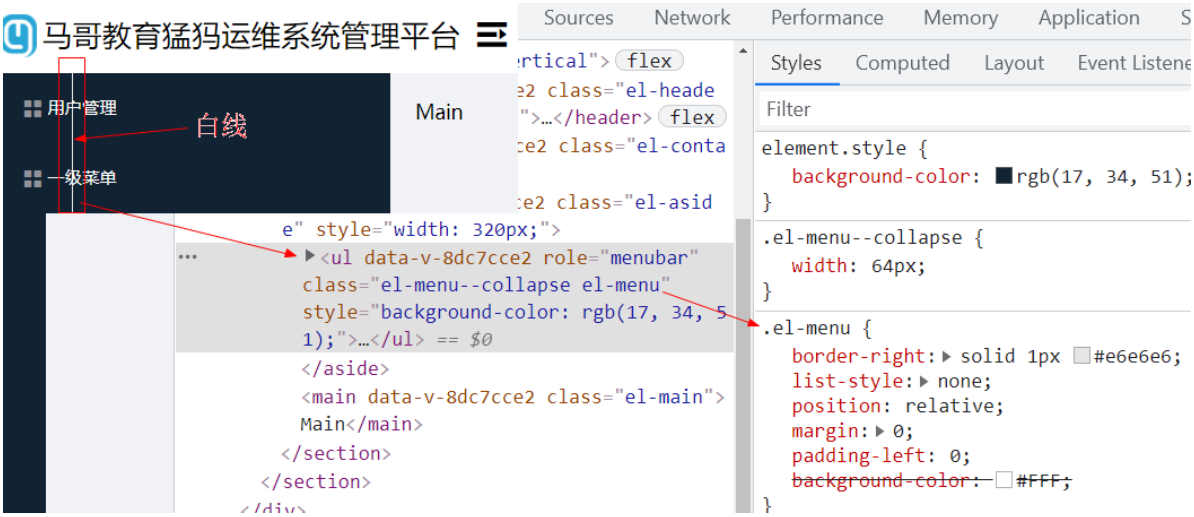
为Menu组件增加一个绑定属性`:collapse="isCollapse"`才能折叠，是否开启动画`:collapse-transition="true"`

Home.vue

```
<template>
  <el-container>
    <el-header>
      <div class="logo">
        <img src="../assets/logo.png" alt="logo" />
        <div class="title">马哥教育猛犸运维系统管理平台</div>
        <div><i :class="isCollapsed ? 'el-icon-s-unfold' : 'el-icon-s-fold'" @click="toggleMenu"></i></div>
      </div>
      <div class="info">
        <el-button type="info" @click="logout">退出</el-button>
      </div>
    </el-header>
    <el-container>
      <el-aside :width="isCollapsed ? '64px' : '200px'">
        <el-menu
          default-active="101"
          background-color="#123"
          text-color="#fff"
          active-text-color="#ffd04b"
          :collapse="isCollapsed"
          :collapse-transition="false"
        >
          <el-submenu v-for="item in menulist" :index="item.id + ''" :key="item.id">
            <template slot="title">
              <i class="el-icon-menu"></i><span slot="title">{{ item.itemName }}</span>
            </template>
            <el-menu-item v-for="subItem in item.children" :index="subItem.id + ''" :key="subItem.id">
              <template slot="title">
                <i class="el-icon-menu"></i><span slot="title">{{ subItem.itemName }}</span>
              </template>
            </el-menu-item>
          </el-submenu>
        </el-menu>
      </el-aside>
      <el-main>Main</el-main>
    </el-container>
  </el-container>
</template>

<script>
export default {
  created() {
    // vue组件创建时查菜单数据
```

```
    this.getMenulist()
  },
  data() {
    return {
      menulist: [], // 菜单数据
      isCollapsed: true
    }
  },
  methods: {
    async getMenulist() {
      const { data: response } = await this.$http.get('users/menulist/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.menulist = response.data // 菜单项数组
    },
    logout() {
      window.localStorage.removeItem('token')
      this.$router.push('/login')
    },
    toggleMenu() {
      this.isCollapsed = !this.isCollapsed
    }
  }
}
</script>

<style lang="less" scoped>
.el-container {
  height: 100%;
}

.el-header {
  display: flex;
  justify-content: space-between; // 主轴两端对齐
  align-items: center; // 侧轴居中
  padding-left: 5px;
  .logo {
    display: flex;
    img {
      width: 30px;
    }
    .title {
      font-size: 24px;
      margin-left: 5px;
    }
    .el-icon-s-unfold,
    .el-icon-s-fold {
      font-size: 32px;
      margin-left: 10px;
    }
  }
}

.el-aside {
  background-color: #123;
}

```

```
.el-main {
  background-color: #f0f2f5;
}

.el-menu {
  border-right: none; // 消除菜单右侧白线
}
</style>
```



嵌套路由

欢迎页

新建src/components/Welcome.vue

```
<template>
  <div>
    <h3>欢迎访问马哥教育猛犸运维平台系统 by wayne</h3>
  </div>
</template>

<script>
export default {

}
</script>

<style>

</style>
```

嵌套路由

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

```
/user/foo/profile                     /user/foo/posts
+------------------+                  +-----------------+
| User             |                  | User            |
| +--------------+ |                  | +-------------+ |
| | Profile      | |  +------------>  | | Posts       | |
| |              | |                  | |             | |
| +--------------+ |                  | +-------------+ |
+------------------+                  +-----------------+
```

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

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Login from '../components/Login.vue'
import Home from '../components/Home.vue'
import Welcome from '../components/Welcome.vue'

Vue.use(VueRouter)

const routes = [
  { path: '/', redirect: '/login' },
  { path: '/login', component: Login },
  {
    path: '/home',
    component: Home,
    redirect: '/welcome', // 访问/home重定向到/welcome，进入子路由
    children: [
      { path: '/welcome', component: Welcome }
    ]
  }
]

const router = new VueRouter({
  routes
})

// 挂载全局导航守卫
router.beforeEach((to, from, next) => { // from从哪里来，to去哪里，next函数跳转
  if (to.path === '/login') {
    next()
  }
  // 读取token
  const token = window.localStorage.getItem('token')
  if (!token) {
    next('/login')
  }
  next()
})

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">`

```
<template>
  <el-container>
    <el-header>
      <div class="logo">
        <img src="../assets/logo.png" alt="logo" />
        <div class="title">马哥教育猛犸运维系统管理平台</div>
        <div><i :class="isCollapsed ? 'el-icon-s-unfold' : 'el-icon-s-fold'" @click="toggleMenu"></i></div>
      </div>
      <div class="info">
        <el-button type="info" @click="logout">退出</el-button>
      </div>
    </el-header>
    <el-container>
      <el-aside :width="isCollapsed ? '64px' : '200px'">
        <el-menu
          default-active="101"
          background-color="#123"
          text-color="#fff"
          active-text-color="#ffd04b"
          :collapse="isCollapsed"
          :collapse-transition="false"
          :router="true"
        >
          <el-submenu v-for="item in menulist" :index="item.id + ''" :key="item.id">
            <template slot="title">
              <i class="el-icon-menu"></i><span slot="title">{{ item.itemName }}</span>
            </template>
            <el-menu-item v-for="subItem in item.children" :index="subItem.path" :key="subItem.id">
              <template slot="title">
                <i class="el-icon-menu"></i><span slot="title">{{ subItem.itemName }}</span>
              </template>
            </el-menu-item>
          </el-submenu>
        </el-menu>
      </el-aside>
      <el-main>
        <!-- 嵌套路由 -->
        <router-view></router-view>
      </el-main>
    </el-container>
  </el-container>
</template>

<script>
export default {
  created() {
```

```
    // vue组件创建时查菜单数据
    this.getMenulist()
  },
  data() {
    return {
      menulist: [], // 菜单数据
      isCollapsed: false
    }
  },
  methods: {
    async getMenulist() {
      const { data: response } = await this.$http.get('users/menulist/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.menulist = response.data // 菜单项数组
    },
    logout() {
      window.localStorage.removeItem('token')
      this.$router.push('/login')
    },
    toggleMenu() {
      this.isCollapsed = !this.isCollapsed
    }
  }
}
</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

```
from django.http import Http404
from django.core.exceptions import PermissionDenied
from rest_framework.views import set_rollback
from rest_framework import exceptions
from rest_framework.views import Response

class MagBaseException(exceptions.APIException):
    """基类定义基本的异常"""
    code = 10000 # code为0表示正常，非0都是错误
    message = '非法请求' # 错误描述

    @classmethod
    def get_message(cls):
        return {'code': cls.code, 'message': cls.message}

class InvalidUsernameOrPassword(MagBaseException):
    code = 1
    message = '用户名或密码错误'
```

```python

class InvalidToken(MagBaseException):
    code = 2
    message = '认证无效，请重新登录'

class NotAuthenticated(MagBaseException):
    code = 3
    message = '未登录，请重新登录'

# code < 100，前端清除token，跳转到登录页

# 内部异常暴露细节
exc_map = {
    'AuthenticationFailed': InvalidUsernameOrPassword,
    'InvalidToken': InvalidToken,          # token过期
    'NotAuthenticated': NotAuthenticated, # 无token登录
}


def global_exception_handler(exc, context):
    """
    全局异常处理

    照抄rest_framework.views.exception_handler，略作修改
    不管什么异常这里统一处理。根据不同类型显示不同的
    为了前端解析方便，这里响应的状态码采用默认的200
    异常对应处理后返回对应的错误码和错误描述
    异常找不到对应就返回缺省
    """
    if isinstance(exc, Http404):
        exc = exceptions.NotFound()
    elif isinstance(exc, PermissionDenied):
        exc = exceptions.PermissionDenied()

    print('异常', '=' * 30)
    print(type(exc), exc.__class__.__name__, exc.__dict__)
    print('=' * 30)

    if isinstance(exc, exceptions.APIException):
        headers = {}
        if getattr(exc, 'auth_header', None):
            headers['WWW-Authenticate'] = exc.auth_header
        if getattr(exc, 'wait', None):
            headers['Retry-After'] = '%d' % exc.wait

        if isinstance(exc.detail, (list, dict)):
            data = exc.detail
        else:
            data = {'detail': exc.detail}

        set_rollback()

        if isinstance(exc, MagBaseException):
            errmsg = exc.get_message()
        else:
            errmsg = exc_map.get(exc.__class__.__name__,
MagBaseException).get_message()
        return Response(errmsg, status=200) # 状态恒为200
```

```
        #return Response(data, status=exc.status_code, headers=headers)

    return None
```

前台src/main.js

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import './plugins/element.js'
import './assets/css/main.css'
import axios from 'axios'

// 请求拦截器，只要发起异步请求都要带上这个token
axios.interceptors.request.use(function(config) {
  // 在发送请求之前做些什么
  config.headers.Authorization = 'Bearer ' +
window.localStorage.getItem('token')
  return config
})
// 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
axios.interceptors.response.use(function(response) {
  // 对响应数据做点什么
  if (response.data.code && response.data.code < 100) {
    router.push('/login')
  } else {
    return response
  }
})
// baseURL指向后台服务
axios.defaults.baseURL = '/api/v1/'
// 为Vue类增加全局属性$http，这样所有组件实例都可以使用该属性了
Vue.prototype.$http = axios

Vue.config.productionTip = false

new Vue({
  router,
  render: h => h(App)
}).$mount('#app')
```

到此，主页开发完成，分别提交前后端代码。