角色和权限

角色分配权限

显示权限树

树节点的选择参考 https://element.eleme.cn/#/zh-CN/component/tree#shu-jie-dian-de-xuan-ze

- `:data="dataList"` 指定遍历的数据源
- `show-checkbox` 复选框
- `node-key="id"` 指定节点key使用id属性，key是节点唯一标识
- `ref="tree"`
- `:props="defaultProps"` 指定树使用的数据的属性名
- `:default-checked-keys="selectedIds"` 默认勾选的key的数组
- `this.$refs.tree.getCheckedKeys()` 获得被选择的复选框

一个角色设置权限首先可以显示所有权限，不管是和该角色关联的或未关联的都要显示，还要把已经关联的权限打勾。用户可以修改，打勾或去勾，最后还要保存打勾的权限。

前台代码

src/plugins/element.js

使用了树型控件，方便勾选

```
import Vue from 'vue'
import {
  Form, FormItem, Input, Button, Message, Container,
  Header, Aside, Main, Menu, MenuItem, Submenu, Breadcrumb,
  BreadcrumbItem, Card, Row, Col, Table, TableColumn,
  Dialog, Pagination, Switch, MessageBox, Tooltip,
  DropdownMenu, DropdownItem, Dropdown, Tree
} from 'element-ui'

Vue.use(Tree)
```

```
        <el-tree
          ref="tree"
          node-key="id"
          :data="permList"
          :props="defaultProps"
          show-checkbox
          :default-checked-keys="selectedIds"
        ></el-tree>
```

- ref="tree"，为了方便找到树
- show-checkbox增加checkbox
- node-key="id"，this.$refs['tree'].getCheckedKeys()时，返回选中节点的id
- :props="defaultProps"，指定显示用的label的属性和子节点的属性

src/components/user/Roles.vue

功能包括显示权限树、设置权限

```
<template>
  <div>
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>权限列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <div class="grid-content bg-purple-dark">
            <el-input placeholder="请输入内容" v-model="search">
              <el-button slot="append" icon="el-icon-search" @click="getList(1)"></el-button>
            </el-input>
          </div>
        </el-col>
        <el-col :span="12">
          <el-button type="primary" @click="addDialogVisible = true">增加角色</el-button>
        </el-col>
      </el-row>
      <el-table :data="dataList" border style="width: 100%">
        <el-table-column type="index" label="序号"> </el-table-column>
        <el-table-column prop="id" label="id"> </el-table-column>
        <el-table-column prop="name" label="名称"> </el-table-column>
        <el-table-column label="操作" #default="{ row }">
          <el-tooltip content="分配权限" effect="light">
            <el-button type="info" icon="el-icon-setting" size="mini" @click="handleSetPerm(row)"></el-button>
          </el-tooltip>
          <el-tooltip content="修改" effect="light">
            <el-button type="success" icon="el-icon-edit" size="mini"></el-button>
          </el-tooltip>
          <el-tooltip content="删除" effect="light">
            <el-button type="danger" icon="el-icon-delete" size="mini"></el-button>
          </el-tooltip>
        </el-table-column>
      </el-table>
      <el-pagination
        @current-change="handleCurrentChange"
        :current-page="pagination.page"
        :page-size="pagination.size"
        layout="total, prev, pager, next, jumper"
        :total="pagination.total"
      >
      </el-pagination>
    </el-card>
    <!-- 赋权对话框 -->
    <el-dialog title="设置权限" :visible.sync="addPermDialogVisible" @close="resetTree">
```

```
      <el-card>
        <el-tree
          ref="tree"
          node-key="id"
          :data="permList"
          :props="defaultProps"
          show-checkbox
          :default-checked-keys="selectedIds"
        ></el-tree>
      </el-card>
      <div slot="footer" class="dialog-footer">
        <el-button @click="addPermDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="addPerm">确 定</el-button>
      </div>
    </el-dialog>
    <!-- 添加对话框 -->
    <el-dialog title="增加" :visible.sync="addDialogVisible" @close="resetForm('add')">
      <el-form :model="addForm" :rules="addRules" ref="add" label-width="100px">
        <el-form-item label="名称" prop="name">
          <el-input v-model="addForm.name"></el-input>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="addDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="add">确 定</el-button>
      </div>
    </el-dialog>
  </div>
</template>

<script>
export default {
  created() {
    this.getList()
  },
  data() {
    return {
      search: '',
      dataList: [],
      pagination: { page: 1, size: 1, total: 0 },
      addDialogVisible: false,
      addForm: {
        name: ''
      },
      addRules: {
        name: [
          { required: true, message: '请输入名称', trigger: 'blur' },
          { min: 1, max: 16, message: '长度在 1 到 16 个字符', trigger: 'blur' }
        ]
      },
      // 赋权
      addPermDialogVisible: false,
      permList: [],
      selectedIds: [],
      defaultProps: {
        children: 'children',
        label: 'name'
      },
      currentRole: {}
    }
  },
  methods: {
    async getList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('users/roles/', {
        params: {
          page,
          search: this.search // 搜索自行实现
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.dataList = response.results
      this.pagination = response.pagination
    },
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`)
      this.getList(val)
    },
    resetForm(name) {
      this.$refs[name].resetFields()
    },
    add() {
      const name = 'add'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          const { data: response } = await this.$http.post('users/roles/',
this.addForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.addDialogVisible = false
          this.resetForm(name)
          this.getList() // 刷新用户列表
        }
      })
    },
    // 赋权
    async handleSetPerm(row) {
      const { id } = row
      const { data: response } = await
this.$http.get(`users/roles/${id}/perms/`)
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.permList = response.allPerms
      this.selectedIds = response.permissions
      this.currentRole = row
      this.addPermDialogVisible = true
    },
    async addPerm() {
      const name = 'tree'
      const permissions = this.$refs[name].getCheckedKeys()
      const { id } = this.currentRole
      // 用patch部分更新权限
      const { data: response } = await
this.$http.patch(`users/roles/${id}/`, {
        permissions
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.addPermDialogVisible = false
    },
    resetTree() {
      this.permList = []
      this.$refs.tree.setCheckedKeys([])
    }
  }
}
</script>

<style>
</style>
```

后台代码

组权限设置 https://docs.djangoproject.com/en/3.2/ref/contrib/auth/#django.contrib.auth.models.Group

```python
# 组权限设置
group.permissions.set([permission_list]) # 最简单，只需要权限id
group.permissions.add(permission, permission, ...)
group.permissions.remove(permission, permission, ...)
group.permissions.clear()
```

但这次上面组权限设置方法用不上

user/views.py

```python
from rest_framework.decorators import api_view, permission_classes, action
from rest_framework.viewsets import ModelViewSet
from django.contrib.auth.models import Group, Permission

class RoleViewSet(ModelViewSet):
    queryset = Group.objects.all()
    serializer_class = GroupSerializer

    @action(detail=True, methods=['GET'], url_path='roles')
    def roles(self, request, pk=None): # /users/mgr/2/roles/ GET
        user = self.get_object()
        data = UserSerializer(user).data
        data['roles'] = [r.get('id') for r in user.groups.values('id')]
        data['allRoles'] = Group.objects.values('id', 'name')
        return Response(data)
```

用户分配角色

前台代码

回到Users.vue，为其增加分配角色按钮，实现分配角色

src/components/user/Users.vue 完整代码如下

```
<template>
  <div>
    <el-breadcrumb separator-class="el-icon-arrow-right">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>用户列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <div class="grid-content bg-purple-dark">
            <el-input placeholder="请输入内容" v-model="search">
              <el-button slot="append" icon="el-icon-search"
@click="getUserList(1)"></el-button>
            </el-input>
          </div>
        </el-col>
        <el-col :span="12">
          <el-button type="primary" @click="addUserDialogVisible = true">增
加用户</el-button>
        </el-col>
      </el-row>
      <el-table :data="userList" border style="width: 100%">
        <el-table-column type="index" label="序号"> </el-table-column>
        <el-table-column prop="username" label="用户名"> </el-table-column>
        <el-table-column prop="is_active" label="激活">
          <template #default="{ row }">
            <el-switch v-if="row.id !== 1" v-model="row.is_active"
@change="handleIsActiveChange(row)"> </el-switch>
          </template>
        </el-table-column>
        <el-table-column prop="is_superuser" label="管理员"> </el-table-
column>
        <el-table-column prop="phone" label="电话"> </el-table-column>
        <el-table-column label="操作">
          <template #default="{ row }">
            <el-tooltip v-if="row.id !== 1" content="分配角色"
effect="light">
              <el-button @click="handleAuthorUser(row)" type="info"
icon="el-icon-user" size="mini"></el-button>
            </el-tooltip>
            <el-tooltip v-if="row.id !== 1" content="修改" effect="light">
              <el-button @click="handleEditUser(row)" type="success"
icon="el-icon-edit" size="mini"></el-button>
            </el-tooltip>
            <el-tooltip v-if="row.id !== 1" content="删除" effect="light">
```

```
       <el-button @click="handleDelUser(row.id)" type="danger" icon="el-icon-delete" size="mini"></el-button>
      </el-tooltip>
     </template>
    </el-table-column>
   </el-table>
   <el-pagination
    @current-change="handleCurrentChange"
    :current-page="pagination.page"
    :page-size="pagination.size"
    layout="total, prev, pager, next, jumper"
    :total="pagination.total"
   >
   </el-pagination>
  </el-card>
  <!-- 分配角色对话框 -->
  <el-dialog title="设置角色" :visible.sync="addRoleDialogVisible" @close="resetTree">
   <el-card>
    <el-tree
     ref="tree"
     node-key="id"
     :data="roleList"
     show-checkbox
     :default-checked-keys="selectedIds"
     :props="{ label: 'name' }"
    ></el-tree>
   </el-card>
   <div slot="footer" class="dialog-footer">
    <el-button @click="addRoleDialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="addRole">确 定</el-button>
   </div>
  </el-dialog>
  <!-- 修改用户对话框 -->
  <el-dialog title="修改" :visible.sync="editUserDialogVisible" @close="resetForm('editUser')">
   <el-form :model="editUserForm" :rules="editUserRules" ref="editUser" label-width="100px">
    <el-form-item label="用户名" prop="username">{{ editUserForm.username }}</el-form-item>
    <el-form-item label="电话" prop="phone">
     <el-input v-model="editUserForm.phone"></el-input>
    </el-form-item>
    <el-form-item label="姓" prop="last_name">
     <el-input v-model="editUserForm.last_name"></el-input>
    </el-form-item>
    <el-form-item label="名" prop="first_name">
     <el-input v-model="editUserForm.first_name"></el-input>
    </el-form-item>
    <el-form-item label="邮箱" prop="email">
     <el-input v-model="editUserForm.email"></el-input>
    </el-form-item>
   </el-form>
   <span slot="footer" class="dialog-footer">
    <el-button @click="editUserDialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="editUser">确 定</el-button>
   </span>
  </el-dialog>
```

```
    <!-- 添加用户对话框 -->
    <el-dialog title="增加" :visible.sync="addUserDialogVisible" @close="resetForm('addUser')">
      <el-form :model="addUserForm" :rules="addUserRules" ref="addUser" label-width="100px">
        <el-form-item label="用户名" prop="username">
          <el-input v-model="addUserForm.username"></el-input>
        </el-form-item>
        <el-form-item label="密码" prop="password">
          <el-input type="password" v-model="addUserForm.password"></el-input>
        </el-form-item>
        <el-form-item label="确认密码" prop="checkPass">
          <el-input type="password" v-model="addUserForm.checkPass"></el-input>
        </el-form-item>
        <el-form-item label="电话" prop="phone">
          <el-input v-model="addUserForm.phone"></el-input>
        </el-form-item>
        <el-form-item label="姓" prop="last_name">
          <el-input v-model="addUserForm.last_name"></el-input>
        </el-form-item>
        <el-form-item label="名" prop="first_name">
          <el-input v-model="addUserForm.first_name"></el-input>
        </el-form-item>
        <el-form-item label="邮箱" prop="email">
          <el-input v-model="addUserForm.email"></el-input>
        </el-form-item>
      </el-form>
      <span slot="footer" class="dialog-footer">
        <el-button @click="addUserDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="addUser">确 定</el-button>
      </span>
    </el-dialog>
  </div>
</template>

<script>
export default {
  created() {
    this.getUserList()
  },
  data() {
    const validatePass = (rule, value, callback) => {
      if (value !== this.addUserForm.password) {
        callback(new Error('两次输入密码不一致!'))
      } else {
        callback()
      }
    }
    return {
      search: '',
      addUserDialogVisible: false,
      addUserForm: {
        username: '',
        password: '',
        phone: '',
        first_name: '',
        last_name: '',
        email: ''
      },
      addUserRules: {
        username: [
          { required: true, message: '请输入登录用户名', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger: 'blur' }
        ],
        password: [
          { required: true, message: '请输入密码', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger: 'blur' }
        ],
        checkPass: [
          { required: true, message: '请输入密码', trigger: 'blur' },
          { validator: validatePass, trigger: 'blur' }
        ]
      },
      userList: [],
      pagination: { page: 1, size: 20, total: 0 },
      // 修改用户对话框
      editUserDialogVisible: false,
      editUserForm: {
        username: '',
        phone: '',
        first_name: '',
        last_name: '',
        email: ''
      },
      editUserRules: {},
      // 分配角色
      addRoleDialogVisible: false,
      roleList: [],
      selectedIds: [],
      currentUser: {}
    }
  },
  methods: {
    resetForm(name) {
      this.$refs[name].resetFields()
    },
    addUser() {
      const name = 'addUser'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          const { data: response } = await this.$http.post('users/mgr/', this.addUserForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.addUserDialogVisible = false
          this.resetForm(name)
          this.getUserList()
        }
      })
    },
    async getUserList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('users/mgr/', {
        params: {
          page,
          search: this.search
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.userList = response.results
      this.pagination = response.pagination
    },
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`)
      this.getUserList(val)
    },
    async handleIsActiveChange(row) {
      // 激活按钮变化，去后台部分更新字段
      await this.$http.patch(`users/mgr/${row.id}/`, { is_active: row.is_active })
    },
    handleEditUser(row) {
      this.editUserForm = row
      this.editUserDialogVisible = true
    },
    editUser() {
      const { id } = this.editUserForm
      const name = 'editUser'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          const { data: response } = await this.$http.patch(`users/mgr/${id}/`, this.editUserForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.editUserDialogVisible = false
          this.resetForm(name)
          this.getUserList(this.pagination.page) // 刷新用户列表
        }
      })
    },
    // 删除用户
    handleDelUser(id) {
      this.$msgbox
        .confirm('删除该用户, 是否继续?', '提示', {
          confirmButtonText: '确定',
          cancelButtonText: '取消',
          type: 'danger',
          center: true
        })
        .then(async () => {
          const { data: response } = await this.$http.delete(`users/mgr/${id}/`)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.getUserList()
        })
        .catch(() => {})
    },
    // 分配角色
    async handleAuthorUser(row) {
      console.log(row)
      const { id } = row
      const { data: response } = await this.$http.get(`users/mgr/${id}/roles/`)
      if (response.code) {
        return this.$message.error(response.message)
      }
      console.log(response)
      this.roleList = response.allRoles
      this.selectedIds = response.roles
      this.currentUser = row
      this.addRoleDialogVisible = true
    },
    async addRole() {
      const name = 'tree'
      const roles = this.$refs[name].getCheckedKeys()
      const { id } = this.currentUser
      // 用patch部分更新权限
      const { data: response } = await this.$http.put(`users/mgr/${id}/roles/`, {
        roles
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.addRoleDialogVisible = false
    },
    resetTree() {
      this.roleList = []
      this.$refs.tree.setCheckedKeys([])
    }
  }
}
</script>

<style lang="less" scoped></style>
```

后台代码

用户设置组 https://docs.djangoproject.com/en/3.2/topics/auth/default/#topic-authorization

```
myuser.groups.set([group_list]) #  # 最简单，只需要组id
myuser.groups.add(group, group, ...)
myuser.groups.remove(group, group, ...)
myuser.groups.clear()
```

添加额外的路由

Routing additional HTTP methods for extra actions https://www.django-rest-framework.org/api-guide/viewsets/#routing-additional-http-methods-for-extra-actions

同一个路由，不同的方法

```python
class UserViewSet(ModelViewSet):
    queryset = get_user_model().objects.all()
    serializer_class = UserSerializer
    # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
    filter_backends = [filters.SearchFilter] # 指定filter
    # filterset_fields = ['username']
    search_fields = ['username']

    # 详情页禁止修改username，如果提供用户名，它就要验证用户名唯一性，且尝试修改用户名
    def partial_update(self, request, *args, **kwargs):
        print('~~~~@@@@@@@@@@@@@@')
        print(request.user)
        print(request.data)
        print('~~~~@@@@@@@@@@@@@@')
        request.data.pop('username', None) # 剔除不要更新的字段
        request.data.pop('id', None)
        request.data.pop('password', None)
        return super().partial_update(request, *args, **kwargs)

    def get_object(self):
        if self.request.method.lower() != 'get':
            pk = self.kwargs.get('pk')
            if pk == 1 or pk == '1':
                print('竟敢修改管理员！！！！')
                raise Http404
        return super().get_object()

    @action(['GET'], detail=False, url_path='whoami')
    def whoami(self, request): # detail=False，非详情页
        print(request.user)
        return Response({
            'user':{
                'id': request.user.id,
                'username': request.user.username
            }
        })

    @action(detail=True, methods=['post'], url_path='setpwd')
    def set_password(self, request, pk=None):
        user = self.get_object()
        # {'oldPassword': 'adminadmin', 'password': 'admin', 'checkPass': 'admin'}
        if user.check_password(request.data['oldPassword']):
            user.set_password(request.data['password'])
            user.save()
            return Response()
        else:
            raise InvalidPassword

    @action(detail=True, methods=['GET'], url_path='roles')
    def roles(self, request, pk=None): # /users/mgr/2/roles/ GET
```

```python
        user = self.get_object()
        data = UserSerializer(user).data
        data['roles'] = [r.get('id') for r in user.groups.values('id')]
        data['allRoles'] = Group.objects.values('id', 'name')
        return Response(data)

    @roles.mapping.put  # /users/mgr/2/roles/ PUT
    def set_roles(self, request, pk=None):
        user = self.get_object()
        roles = request.data.get('roles', [])
        user.groups.set(roles)
        print(user.groups.all())
        return Response(status=201)
```

DRF权限

has_perm、has_perms参考 https://docs.djangoproject.com/en/3.2/ref/contrib/auth/#django.contrib.auth.models.User.has_perm

`<app_label>.<permission_codename>`是权限perm的书写方式。

超级用户拥有所有权限

```python
from user.models import UserProfile

user = UserProfile.objects.get(pk=2)
print(user.groups.all())
print(user.is_superuser, user.is_staff, user.is_active)
print(user.has_perm('user.view_userprofile'))
print(user.has_perms(['user.view_userprofile']))
print(user.has_perms([])) # 特别特别特别注意，这个返回True
```

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

DRF可以为每一个ViewSet、View设置权限。

在APIView的`dispatch()`中调用了`initial()`，`initial()`中又调用了`check_permissions()`。可以看出，如果任意一个给出的权限不具备，立即返回`permission_denied`

AllowAny

AllowAny是默认权限，不管是否登录成功都可以访问。

IsAuthenticated

```python
class IsAuthenticated(BasePermission):
    """
    Allows access only to authenticated users.
    """

    def has_permission(self, request, view):
        return bool(request.user and request.user.is_authenticated)
```

rest_framework.permissions.IsAuthenticated根本没有查权限表，就是判断是否登录，如果登录了就行。

IsAdminUser

```
class IsAdminUser(BasePermission):
    """
    Allows access only to admin users.
    """

    def has_permission(self, request, view):
        return bool(request.user and request.user.is_staff)
```

DjangoModelPermissions

此权限类与 Django 的标准django.contrib.auth 模型权限相关联。此权限只能应用于具有.queryset属性或get_queryset()方法的视图类。仅当用户通过身份验证并分配了相关模型权限时，才会授予权限。

- POST请求要求用户拥有模型的add权限
- PUT和PATCH请求要求用户拥有模型的change权限
- DELETE请求要求用户拥有模型的delete权限

也可以覆盖默认行为以支持自定义模型权限。

GET方法并没有被限制，而我们需要，所以一定要自定义。

要使用自定义模型权限，请覆盖DjangoModelPermissions并设置该.perms_map属性。

utils/permissions.py

```
from rest_framework.permissions import DjangoModelPermissions

class CrudModelPermission(DjangoModelPermissions):
    # 和应用的Model类的增删改查对应
    perms_map = {
        'GET': ['%(app_label)s.view_%(model_name)s'], # 增加view查看权限GET
        'POST': ['%(app_label)s.add_%(model_name)s'],
        'PUT': ['%(app_label)s.change_%(model_name)s'],
        'PATCH': ['%(app_label)s.change_%(model_name)s'],
        'DELETE': ['%(app_label)s.delete_%(model_name)s'],
    }
```

全局设定权限

```
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
        'utils.permissions.CrudModelPermission'
    ],
    'DEFAULT_PAGINATION_CLASS': 'utils.paginations.PageNumberPagination',
    'PAGE_SIZE': 20,
    'DEFAULT_FILTER_BACKENDS':
['django_filters.rest_framework.DjangoFilterBackend'],
}
```

user/views.py完整代码如下

```
from rest_framework.views import Response, Request
from rest_framework.decorators import api_view, permission_classes, action
from rest_framework.permissions import IsAuthenticated, IsAdminUser
from rest_framework.viewsets import ModelViewSet, ReadOnlyModelViewSet
from django.contrib.auth import get_user_model
from .models import UserProfile
from .serializers import UserSerializer, PermSerializer, GroupSerializer
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework import filters
from django.http.response import Http404
from utils.exceptions import InvalidPassword
from django.contrib.auth.models import Group, Permission, ContentType
from utils.permissions import CrudModelPermission

class RoleViewSet(ModelViewSet):
    queryset = Group.objects.all()
    serializer_class = GroupSerializer

    @action(['GET'], detail=True, url_path='perms')
    def perms(self, request, pk):
        obj = self.get_object()
        data = GroupSerializer(obj).data # 当前角色和其权限ids
        data['allPerms'] = list(PermViewSet.queryset.values('id', 'name'))
        return Response(data)

_exclude_contenttypes = [c.id for c in
ContentType.objects.filter(model__in=[
    'logentry', 'group', 'permission',
    'contenttype', 'session'
])] # 排除掉Django内建应用的权限
# print(_exclude_contenttypes, '+++++++++++++++++++')

class PermViewSet(ReadOnlyModelViewSet):
    queryset =
Permission.objects.exclude(content_type__in=_exclude_contenttypes)
    serializer_class = PermSerializer
    filter_backends = [filters.SearchFilter]
    search_fields = ['name', 'codename']
```

```python


class UserViewSet(ModelViewSet):
    queryset = get_user_model().objects.all()
    serializer_class = UserSerializer
    # permission_classes = [CrudModelPermission] # 必须有UserProfile的增删改查权限
    filter_backends = [filters.SearchFilter] # 指定filter
    # filterset_fields = ['username']
    search_fields = ['username']

    # 详情页禁止修改username，如果提供用户名，它就要验证用户名唯一性，且尝试修改用户名
    def partial_update(self, request, *args, **kwargs):
        print('~~~~@@@@@@@@@@@@@')
        print(request.user)
        print(request.data)
        print('~~~~@@@@@@@@@@@@@')
        request.data.pop('username', None) # 剔除不要更新的字段
        request.data.pop('id', None)
        request.data.pop('password', None)
        return super().partial_update(request, *args, **kwargs)

    def get_object(self):
        if self.request.method.lower() != 'get':
            pk = self.kwargs.get('pk')
            if pk == 1 or pk == '1':
                print('竟敢修改管理员！！！！')
                raise Http404
        return super().get_object()

    @action(['GET'], detail=False, url_path='whoami')
    def whoami(self, request): # detail=False，非详情页
        print(request.user)
        return Response({
            'user':{
                'id': request.user.id,
                'username': request.user.username
            }
        })

    @action(detail=True, methods=['post'], url_path='setpwd')
    def set_password(self, request, pk=None):
        user = self.get_object()
        # {'oldPassword': 'adminadmin', 'password': 'admin', 'checkPass': 'admin'}
        if user.check_password(request.data['oldPassword']):
            user.set_password(request.data['password'])
            user.save()
            return Response()
        else:
            raise InvalidPassword

    @action(detail=True, methods=['GET'], url_path='roles')
    def roles(self, request, pk=None): # /users/mgr/2/roles/ GET
        user = self.get_object()
        data = UserSerializer(user).data
        data['roles'] = [r.get('id') for r in user.groups.values('id')]
        data['allRoles'] = Group.objects.values('id', 'name')
```

```python
        return Response(data)

    @roles.mapping.put  # /users/mgr/2/roles/ PUT
    def set_roles(self, request, pk=None):
        user = self.get_object()
        roles = request.data.get('roles', [])
        user.groups.set(roles)
        print(user.groups.all())
        return Response(status=201)


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):
        self[key] = value

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

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

from django.contrib.auth.models import AbstractUser
@api_view(['GET'])
@permission_classes([IsAuthenticated]) # 覆盖全局配置单独设置权限
def menulist_view(request:Request):
    user:AbstractUser = 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)

    # 权限测试
    if user.has_perm('user.view_userprofile'):
        p9 = PermissionItem(901, '管理')
        p9.append(PermissionItem(902, 'x列表', '/users/'))
        p9.append(PermissionItem(903, 'y列表', '/users/roles/'))
        p9.append(PermissionItem(904, 'z列表', '/users/perms/'))
        menulist['data'].append(p9)
    # 普通用户测试
    p10 = PermissionItem(1001, '测试项')
    p10.append(PermissionItem(1002, '测试用', '/test'))
    menulist['data'].append(p10)
    return Response(menulist)
```

使用权限，灵活控制各模块权限