

# 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

```
1 import Vue from 'vue'
2 import {
3   Form, FormItem, Input, Button, Message, Container,
4   Header, Aside, Main, Menu, MenuItem, Submenu, Breadcrumb,
5   BreadcrumbItem, Card, Row, Col, Table, TableColumn,
6   Dialog, Pagination, Switch, MessageBox, Tooltip,
7   DropdownMenu, DropdownItem, Dropdown, Tree
8 } from 'element-ui'
9
10 Vue.use(Tree)
```

```
1 <el-tree
2   ref="tree"
3   node-key="id"
4   :data="permList"
5   :props="defaultProps"
6   show-checkbox
7   :default-checked-keys="selectedIds"
8 ></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

功能包括显示权限树、设置权限

```

1  <template>
2    <div>
3      <el-breadcrumb separator-class="el-icon-arrow-right">
4        <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5        <el-breadcrumb-item>用户管理</el-breadcrumb-item>
6        <el-breadcrumb-item>权限列表</el-breadcrumb-item>
7      </el-breadcrumb>
8      <el-card class="box-card">
9        <el-row :gutter="20">
10         <el-col :span="12">
11           <div class="grid-content bg-purple-dark">
12             <el-input placeholder="请输入内容" v-model="search">
13               <el-button slot="append" icon="el-icon-search"
14                 @click="getList(1)"></el-button>
15             </el-input>
16           </div>
17         </el-col>
18         <el-col :span="12">
19           <el-button type="primary" @click="addDialogVisible = true">增加角
20             色</el-button>
21         </el-col>
22       </el-row>
23       <el-table :data="dataList" border style="width: 100%">
24         <el-table-column type="index" label="序号"> </el-table-column>
25         <el-table-column prop="id" label="id"> </el-table-column>
26         <el-table-column prop="name" label="名称"> </el-table-column>
27         <el-table-column label="操作" #default="{ row }">
28           <el-tooltip content="分配权限" effect="light">
29             <el-button type="info" icon="el-icon-setting" size="mini"
30               @click="handleSetPerm(row)"></el-button>
31           </el-tooltip>
32           <el-tooltip content="修改" effect="light">
33             <el-button type="success" icon="el-icon-edit" size="mini"></el-
34             button>
35           </el-tooltip>
36           <el-tooltip content="删除" effect="light">
37             <el-button type="danger" icon="el-icon-delete" size="mini">
38             </el-button>
39           </el-tooltip>
40         </el-table-column>
41       </el-table>
42       <el-pagination
43         @current-change="handleCurrentChange"
44         :current-page="pagination.page"
45         :page-size="pagination.size"
46         layout="total, prev, pager, next, jumper"
47         :total="pagination.total"
48       >
49     </el-pagination>
50   </el-card>
51   <!-- 赋权对话框 -->
52   <el-dialog title="设置权限" :visible.sync="addPermDialogVisible"
53     @close="resetTree">

```

```

48     <el-card>
49       <el-tree
50         ref="tree"
51         node-key="id"
52         :data="permList"
53         :props="defaultProps"
54         show-checkbox
55         :default-checked-keys="selectedIds"
56       ></el-tree>
57     </el-card>
58     <div slot="footer" class="dialog-footer">
59       <el-button @click="addPermDialogVisible = false">取 消</el-button>
60       <el-button type="primary" @click="addPerm">确 定</el-button>
61     </div>
62   </el-dialog>
63   <!-- 添加对话框 -->
64   <el-dialog title="增加" :visible.sync="addDialogVisible"
@close="resetForm('add')">
65     <el-form :model="addForm" :rules="addRules" ref="add" label-
width="100px">
66       <el-form-item label="名称" prop="name">
67         <el-input v-model="addForm.name"></el-input>
68       </el-form-item>
69     </el-form>
70     <div slot="footer" class="dialog-footer">
71       <el-button @click="addDialogVisible = false">取 消</el-button>
72       <el-button type="primary" @click="add">确 定</el-button>
73     </div>
74   </el-dialog>
75 </div>
76 </template>
77
78 <script>
79 export default {
80   created() {
81     this.getList()
82   },
83   data() {
84     return {
85       search: '',
86       dataList: [],
87       pagination: { page: 1, size: 1, total: 0 },
88       addDialogVisible: false,
89       addForm: {
90         name: ''
91       },
92       addRules: {
93         name: [
94           { required: true, message: '请输入名称', trigger: 'blur' },
95           { min: 1, max: 16, message: '长度在 1 到 16 个字符', trigger:
'blur' }
96         ]
97       },
98       // 赋权
99       addPermDialogVisible: false,
100       permList: [],
101       selectedIds: [],
102       defaultProps: {

```

```

103     children: 'children',
104     label: 'name'
105   },
106   currentRole: {}
107 }
108 },
109 methods: {
110   async getList(page = 1) {
111     if (!page) {
112       page = 1
113     }
114     const { data: response } = await this.$http.get('users/roles/', {
115       params: {
116         page,
117         search: this.search // 搜索自行实现
118       }
119     })
120     if (response.code) {
121       return this.$message.error(response.message)
122     }
123     this.dataList = response.results
124     this.pagination = response.pagination
125   },
126   handleCurrentChange(val) {
127     console.log(`当前页: ${val}`)
128     this.getList(val)
129   },
130   resetForm(name) {
131     this.$refs[name].resetFields()
132   },
133   add() {
134     const name = 'add'
135     this.$refs[name].validate(async (valid) => {
136       if (valid) {
137         const { data: response } = await this.$http.post('users/roles/',
this.addForm)
138         if (response.code) {
139           return this.$message.error(response.message)
140         }
141         this.addDialogVisible = false
142         this.resetForm(name)
143         this.getList() // 刷新用户列表
144       }
145     })
146   },
147   // 赋权
148   async handleSetPerm(row) {
149     const { id } = row
150     const { data: response } = await
this.$http.get(`users/roles/${id}/perms/`)
151     if (response.code) {
152       return this.$message.error(response.message)
153     }
154     this.permList = response.allPerms
155     this.selectedIds = response.permissions
156     this.currentRole = row
157     this.addPermDialogVisible = true
158   },

```

```

159     async addPerm() {
160         const name = 'tree'
161         const permissions = this.$refs[name].getCheckedKeys()
162         const { id } = this.currentRole
163         // 用patch部分更新权限
164         const { data: response } = await
165         this.$http.patch(`users/roles/${id}/`, {
166             permissions
167         })
168         if (response.code) {
169             return this.$message.error(response.message)
170         }
171         this.addPermDialogVisible = false
172     },
173     resetTree() {
174         this.permList = []
175         this.$refs.tree.setCheckedKeys([])
176     }
177 }
178 </script>
179
180 <style>
181 </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>

```

1  # 组权限设置
2  group.permissions.set([permission_list]) # 最简单，只需要权限id
3  group.permissions.add(permission, permission, ...)
4  group.permissions.remove(permission, permission, ...)
5  group.permissions.clear()

```

但这次上面组权限设置方法用不上

user/views.py

```

1  from rest_framework.decorators import api_view, permission_classes, action
2  from rest_framework.viewsets import ModelViewSet
3  from django.contrib.auth.models import Group, Permission
4
5  class RoleViewSet(ModelViewSet):
6      queryset = Group.objects.all()
7      serializer_class = GroupSerializer
8
9      @action(detail=True, methods=['GET'], url_path='roles')
10     def roles(self, request, pk=None): # /users/mgr/2/roles/ GET
11         user = self.get_object()
12         data = UserSerializer(user).data
13         data['roles'] = [r.get('id') for r in user.groups.values('id')]
14         data['allRoles'] = Group.objects.values('id', 'name')
15         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

```
1 <template>
2   <div>
3     <el-breadcrumb separator-class="el-icon-arrow-right">
4       <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>用户管理</el-breadcrumb-item>
6       <el-breadcrumb-item>用户列表</el-breadcrumb-item>
7     </el-breadcrumb>
8     <el-card class="box-card">
9       <el-row :gutter="20">
10        <el-col :span="12">
11          <div class="grid-content bg-purple-dark">
12            <el-input placeholder="请输入内容" v-model="search">
13              <el-button slot="append" icon="el-icon-search"
14                @click="getUserList(1)"></el-button>
15            </el-input>
16          </div>
17        </el-col>
18        <el-col :span="12">
19          <el-button type="primary" @click="addUserDialogVisible = true">增
20            加用户</el-button>
21        </el-col>
22      </el-row>
23      <el-table :data="userList" border style="width: 100%">
24        <el-table-column type="index" label="序号"> </el-table-column>
25        <el-table-column prop="username" label="用户名"> </el-table-column>
26        <el-table-column prop="is_active" label="激活">
27          <template #default="{ row }">
28            <el-switch v-if="row.id !== 1" v-model="row.is_active"
29              @change="handleIsActiveChange(row)"> </el-switch>
30          </template>
31        </el-table-column>
32        <el-table-column prop="is_superuser" label="管理员"> </el-table-
33        column>
34        <el-table-column prop="phone" label="电话"> </el-table-column>
35        <el-table-column label="操作">
36          <template #default="{ row }">
37            <el-tooltip v-if="row.id !== 1" content="分配角色"
38              effect="light">
39              <el-button @click="handleAuthorUser(row)" type="info"
40                icon="el-icon-user" size="mini"></el-button>
41            </el-tooltip>
42            <el-tooltip v-if="row.id !== 1" content="修改" effect="light">
43              <el-button @click="handleEditUser(row)" type="success"
44                icon="el-icon-edit" size="mini"></el-button>
45            </el-tooltip>
46            <el-tooltip v-if="row.id !== 1" content="删除" effect="light">
```

```

40         <el-button @click="handleDelUser(row.id)" type="danger"
icon="el-icon-delete" size="mini"></el-button>
41     </el-tooltip>
42 </template>
43 </el-table-column>
44 </el-table>
45 <el-pagination
46     @current-change="handleCurrentChange"
47     :current-page="pagination.page"
48     :page-size="pagination.size"
49     layout="total, prev, pager, next, jumper"
50     :total="pagination.total"
51 >
52 </el-pagination>
53 </el-card>
54 <!-- 分配角色对话框 -->
55 <el-dialog title="设置角色" :visible.sync="addRoleDialogVisible"
@close="resetTree">
56     <el-card>
57         <el-tree
58             ref="tree"
59             node-key="id"
60             :data="roleList"
61             show-checkbox
62             :default-checked-keys="selectedIds"
63             :props="{ label: 'name' }"
64         ></el-tree>
65     </el-card>
66     <div slot="footer" class="dialog-footer">
67         <el-button @click="addRoleDialogVisible = false">取 消</el-button>
68         <el-button type="primary" @click="addRole">确 定</el-button>
69     </div>
70 </el-dialog>
71 <!-- 修改用户对话框 -->
72 <el-dialog title="修改" :visible.sync="editUserDialogVisible"
@close="resetForm('editUser')">
73     <el-form :model="editUserForm" :rules="editUserRules" ref="editUser"
label-width="100px">
74         <el-form-item label="用户名" prop="username">{{
editUserForm.username }}</el-form-item>
75         <el-form-item label="电话" prop="phone">
76             <el-input v-model="editUserForm.phone"></el-input>
77         </el-form-item>
78         <el-form-item label="姓" prop="last_name">
79             <el-input v-model="editUserForm.last_name"></el-input>
80         </el-form-item>
81         <el-form-item label="名" prop="first_name">
82             <el-input v-model="editUserForm.first_name"></el-input>
83         </el-form-item>
84         <el-form-item label="邮箱" prop="email">
85             <el-input v-model="editUserForm.email"></el-input>
86         </el-form-item>
87     </el-form>
88     <span slot="footer" class="dialog-footer">
89         <el-button @click="editUserDialogVisible = false">取 消</el-button>
90         <el-button type="primary" @click="editUser">确 定</el-button>
91     </span>
92 </el-dialog>

```

```

93     <!-- 添加用户对话框 -->
94     <el-dialog title="增加" :visible.sync="addUserDialogVisible"
@close="resetForm('addUser')">
95         <el-form :model="addUserForm" :rules="addUserRules" ref="addUser"
label-width="100px">
96             <el-form-item label="用户名" prop="username">
97                 <el-input v-model="addUserForm.username"></el-input>
98             </el-form-item>
99             <el-form-item label="密码" prop="password">
100                 <el-input type="password" v-model="addUserForm.password"></el-
input>
101             </el-form-item>
102             <el-form-item label="确认密码" prop="checkPass">
103                 <el-input type="password" v-model="addUserForm.checkPass"></el-
input>
104             </el-form-item>
105             <el-form-item label="电话" prop="phone">
106                 <el-input v-model="addUserForm.phone"></el-input>
107             </el-form-item>
108             <el-form-item label="姓" prop="last_name">
109                 <el-input v-model="addUserForm.last_name"></el-input>
110             </el-form-item>
111             <el-form-item label="名" prop="first_name">
112                 <el-input v-model="addUserForm.first_name"></el-input>
113             </el-form-item>
114             <el-form-item label="邮箱" prop="email">
115                 <el-input v-model="addUserForm.email"></el-input>
116             </el-form-item>
117         </el-form>
118         <span slot="footer" class="dialog-footer">
119             <el-button @click="addUserDialogVisible = false">取 消</el-button>
120             <el-button type="primary" @click="addUser">确 定</el-button>
121         </span>
122     </el-dialog>
123 </div>
124 </template>
125
126 <script>
127 export default {
128     created() {
129         this.getUserList()
130     },
131     data() {
132         const validatePass = (rule, value, callback) => {
133             if (value !== this.addUserForm.password) {
134                 callback(new Error('两次输入密码不一致!'))
135             } else {
136                 callback()
137             }
138         }
139         return {
140             search: '',
141             addUserDialogVisible: false,
142             addUserForm: {
143                 username: '',
144                 password: '',
145                 phone: '',
146                 first_name: '',

```



```

147         last_name: '',
148         email: ''
149     },
150     addUserRules: {
151         username: [
152             { required: true, message: '请输入登录用户名', trigger: 'blur' },
153             { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
154         ],
155         password: [
156             { required: true, message: '请输入密码', trigger: 'blur' },
157             { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
158         ],
159         checkPass: [
160             { required: true, message: '请输入密码', trigger: 'blur' },
161             { validator: validatePass, trigger: 'blur' }
162         ]
163     },
164     userList: [],
165     pagination: { page: 1, size: 20, total: 0 },
166     // 修改用户对话框
167     editUserDialogVisible: false,
168     editUserForm: {
169         username: '',
170         phone: '',
171         first_name: '',
172         last_name: '',
173         email: ''
174     },
175     editUserRules: {},
176     // 分配角色
177     addRoleDialogVisible: false,
178     roleList: [],
179     selectedIds: [],
180     currentUser: {}
181 }
182 },
183 methods: {
184     resetForm(name) {
185         this.$refs[name].resetFields()
186     },
187     addUser() {
188         const name = 'addUser'
189         this.$refs[name].validate(async (valid) => {
190             if (valid) {
191                 const { data: response } = await this.$http.post('users/mgr/',
this.addUserForm)
192                 if (response.code) {
193                     return this.$message.error(response.message)
194                 }
195                 this.addUserDialogVisible = false
196                 this.resetForm(name)
197                 this.getUserList()
198             }
199         })
200     },
201     async getUserList(page = 1) {

```

```

202     if (!page) {
203         page = 1
204     }
205     const { data: response } = await this.$http.get('users/mgr/', {
206         params: {
207             page,
208             search: this.search
209         }
210     })
211     if (response.code) {
212         return this.$message.error(response.message)
213     }
214     this.userList = response.results
215     this.pagination = response.pagination
216 },
217 handleCurrentChange(val) {
218     console.log(`当前页: ${val}`)
219     this.getUserList(val)
220 },
221 async handleIsActiveChange(row) {
222     // 激活按钮变化, 去后台部分更新字段
223     await this.$http.patch(`users/mgr/${row.id}/`, { is_active:
row.is_active })
224 },
225 handleEditUser(row) {
226     this.editUserForm = row
227     this.editUserDialogVisible = true
228 },
229 editUser() {
230     const { id } = this.editUserForm
231     const name = 'editUser'
232     this.$refs[name].validate(async (valid) => {
233         if (valid) {
234             const { data: response } = await
this.$http.patch(`users/mgr/${id}/`, this.editUserForm)
235             if (response.code) {
236                 return this.$message.error(response.message)
237             }
238             this.editUserDialogVisible = false
239             this.resetForm(name)
240             this.getUserList(this.pagination.page) // 刷新用户列表
241         }
242     })
243 },
244 // 删除用户
245 handleDelUser(id) {
246     this.$msgbox
247         .confirm('删除该用户, 是否继续?', '提示', {
248             confirmButtonText: '确定',
249             cancelButtonText: '取消',
250             type: 'danger',
251             center: true
252         })
253         .then(async () => {
254             const { data: response } = await
this.$http.delete(`users/mgr/${id}/`)
255             if (response.code) {
256                 return this.$message.error(response.message)

```

```

257         }
258         this.getUserList()
259     })
260     .catch(() => {})
261 },
262 // 分配角色
263 async handleAuthorUser(row) {
264     console.log(row)
265     const { id } = row
266     const { data: response } = await
this.$http.get(`users/mgr/${id}/roles/`)
267     if (response.code) {
268         return this.$message.error(response.message)
269     }
270     console.log(response)
271     this.roleList = response.allRoles
272     this.selectedIds = response.roles
273     this.currentUser = row
274     this.addRoleDialogVisible = true
275 },
276 async addRole() {
277     const name = 'tree'
278     const roles = this.$refs[name].getCheckedKeys()
279     const { id } = this.currentUser
280     // 用patch部分更新权限
281     const { data: response } = await
this.$http.put(`users/mgr/${id}/roles/`, {
282         roles
283     })
284     if (response.code) {
285         return this.$message.error(response.message)
286     }
287     this.addRoleDialogVisible = false
288 },
289 resetTree() {
290     this.roleList = []
291     this.$refs.tree.setCheckedKeys([])
292 }
293 }
294 }
295 </script>
296
297 <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>

```

1 myuser.groups.set([group_list]) # # 最简单, 只需要组id
2 myuser.groups.add(group, group, ...)
3 myuser.groups.remove(group, group, ...)
4 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

```
1 class UserViewSet(ModelViewSet):
2     queryset = get_user_model().objects.all()
3     serializer_class = UserSerializer
4     # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
5     filter_backends = [filters.SearchFilter] # 指定filter
6     # filterset_fields = ['username']
7     search_fields = ['username']
8
9     # 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
10    def partial_update(self, request, *args, **kwargs):
11        print('~~~~@@@@@@@@@@@@')
12        print(request.user)
13        print(request.data)
14        print('~~~~@@@@@@@@@@@@')
15        request.data.pop('username', None) # 剔除不要更新的字段
16        request.data.pop('id', None)
17        request.data.pop('password', None)
18        return super().partial_update(request, *args, **kwargs)
19
20    def get_object(self):
21        if self.request.method.lower() != 'get':
22            pk = self.kwargs.get('pk')
23            if pk == 1 or pk == '1':
24                print('竟敢修改管理员!!!!')
25                raise Http404
26        return super().get_object()
27
28    @action(['GET'], detail=False, url_path='whoami')
29    def whoami(self, request): # detail=False, 非详情页
30        print(request.user)
31        return Response({
32            'user': {
33                'id': request.user.id,
34                'username': request.user.username
35            }
36        })
37
38    @action(detail=True, methods=['post'], url_path='setpwd')
39    def set_password(self, request, pk=None):
40        user = self.get_object()
41        # {'oldPassword': 'adminadmin', 'password': 'admin', 'checkPass':
'admin'}
42        if user.check_password(request.data['oldPassword']):
43            user.set_password(request.data['password'])
44            user.save()
45            return Response()
46        else:
47            raise InvalidPassword
48
49    @action(detail=True, methods=['GET'], url_path='roles')
50    def roles(self, request, pk=None): # /users/mgr/2/roles/ GET
```

```

51         user = self.get_object()
52         data = UserSerializer(user).data
53         data['roles'] = [r.get('id') for r in user.groups.values('id')]
54         data['allRoles'] = Group.objects.values('id', 'name')
55         return Response(data)
56
57     @roles.mapping.put # /users/mgr/2/roles/ PUT
58     def set_roles(self, request, pk=None):
59         user = self.get_object()
60         roles = request.data.get('roles', [])
61         user.groups.set(roles)
62         print(user.groups.all())
63         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](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

```

1 from user.models import UserProfile
2
3 user = UserProfile.objects.get(pk=2)
4 print(user.groups.all())
5 print(user.is_superuser, user.is_staff, user.is_active)
6 print(user.has_perm('user.view_userprofile'))
7 print(user.has_perms(['user.view_userprofile']))
8 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

```

1 class IsAuthenticated(BasePermission):
2     """
3     Allows access only to authenticated users.
4     """
5
6     def has_permission(self, request, view):
7         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

```
1 class IsAdminUser(BasePermission):
2     """
3     Allows access only to admin users.
4     """
5
6     def has_permission(self, request, view):
7         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

```
1 from rest_framework.permissions import DjangoModelPermissions
2
3 class CrudModelPermission(DjangoModelPermissions):
4     # 和应用的Model类的增删改查对应
5     perms_map = {
6         'GET': ['%(app_label)s.view_%(model_name)s'], # 增加view查看权限GET
7         'POST': ['%(app_label)s.add_%(model_name)s'],
8         'PUT': ['%(app_label)s.change_%(model_name)s'],
9         'PATCH': ['%(app_label)s.change_%(model_name)s'],
10        'DELETE': ['%(app_label)s.delete_%(model_name)s'],
11    }
```

全局设定权限

```

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         'utils.permissions.CrudModelPermission'
9     ],
10    'DEFAULT_PAGINATION_CLASS': 'utils.paginations.PageNumberPagination',
11    'PAGE_SIZE': 20,
12    'DEFAULT_FILTER_BACKENDS':
13    ['django_filters.rest_framework.DjangoFilterBackend'],
14 }

```

user/views.py完整代码如下

```

1 from rest_framework.views import Response, Request
2 from rest_framework.decorators import api_view, permission_classes, action
3 from rest_framework.permissions import IsAuthenticated, IsAdminUser
4 from rest_framework.viewsets import ModelViewSet, ReadOnlyModelViewSet
5 from django.contrib.auth import get_user_model
6 from .models import UserProfile
7 from .serializers import UserSerializer, PermSerializer, GroupSerializer
8 from django_filters.rest_framework import DjangoFilterBackend
9 from rest_framework import filters
10 from django.http.response import Http404
11 from utils.exceptions import InvalidPassword
12 from django.contrib.auth.models import Group, Permission, ContentType
13 from utils.permissions import CrudModelPermission
14
15 class RoleViewSet(ModelViewSet):
16     queryset = Group.objects.all()
17     serializer_class = GroupSerializer
18
19     @action(['GET'], detail=True, url_path='perms')
20     def perms(self, request, pk):
21         obj = self.get_object()
22         data = GroupSerializer(obj).data # 当前角色和其权限ids
23         data['allPerms'] = list(PermViewSet.queryset.values('id', 'name'))
24         return Response(data)
25
26 _exclude_contenttypes = [c.id for c in
27 ContentType.objects.filter(model__in=[
28     'logentry', 'group', 'permission',
29     'contenttype', 'session'
30 ])] # 排除掉Django内建应用的权限
31 # print(_exclude_contenttypes, '+++++')
32
33 class PermViewSet(ReadOnlyModelViewSet):
34     queryset =
35     Permission.objects.exclude(content_type__in=_exclude_contenttypes)
36     serializer_class = PermSerializer
37     filter_backends = [filters.SearchFilter]
38     search_fields = ['name', 'codename']

```

```

37
38
39 class UserViewSet(ModelViewSet):
40     queryset = get_user_model().objects.all()
41     serializer_class = UserSerializer
42     # permission_classes = [CrudModelPermission] # 必须有UserProfile的增删改
查权限
43     filter_backends = [filters.SearchFilter] # 指定filter
44     # filterset_fields = ['username']
45     search_fields = ['username']
46
47     # 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
48     def partial_update(self, request, *args, **kwargs):
49         print('~~~~@@@@@@@@@@@@')
50         print(request.user)
51         print(request.data)
52         print('~~~~@@@@@@@@@@@@')
53         request.data.pop('username', None) # 剔除不要更新的字段
54         request.data.pop('id', None)
55         request.data.pop('password', None)
56         return super().partial_update(request, *args, **kwargs)
57
58     def get_object(self):
59         if self.request.method.lower() != 'get':
60             pk = self.kwargs.get('pk')
61             if pk == 1 or pk == '1':
62                 print('竟敢修改管理员!!!!')
63                 raise Http404
64             return super().get_object()
65
66     @action(['GET'], detail=False, url_path='whoami')
67     def whoami(self, request): # detail=False, 非详情页
68         print(request.user)
69         return Response({
70             'user': {
71                 'id': request.user.id,
72                 'username': request.user.username
73             }
74         })
75
76     @action(detail=True, methods=['post'], url_path='setpwd')
77     def set_password(self, request, pk=None):
78         user = self.get_object()
79         # {'oldPassword': 'adminadmin', 'password': 'admin', 'checkPass':
'admin'}
80         if user.check_password(request.data['oldPassword']):
81             user.set_password(request.data['password'])
82             user.save()
83             return Response()
84         else:
85             raise InvalidPassword
86
87     @action(detail=True, methods=['GET'], url_path='roles')
88     def roles(self, request, pk=None): # /users/mgr/2/roles/ GET
89         user = self.get_object()
90         data = UserSerializer(user).data
91         data['roles'] = [r.get('id') for r in user.groups.values('id')]
92         data['allRoles'] = Group.objects.values('id', 'name')

```



```

93         return Response(data)
94
95     @roles.mapping.put # /users/mgr/2/roles/ PUT
96     def set_roles(self, request, pk=None):
97         user = self.get_object()
98         roles = request.data.get('roles', [])
99         user.groups.set(roles)
100        print(user.groups.all())
101        return Response(status=201)
102
103
104    class PermissionItem(dict):
105        def __init__(self, id, itemName, path=None):
106            super().__init__()
107            self.id = id
108            self.itemName = itemName
109            self.path = path
110            self.children = []
111
112        def __setattr__(self, key, value):
113            self[key] = value
114
115        def __getattr__(self, key):
116            return self[key]
117
118        def append(self, child):
119            self.children.append(child)
120
121    from django.contrib.auth.models import AbstractUser
122    @api_view(['GET'])
123    @permission_classes([IsAuthenticated]) # 覆盖全局配置单独设置权限
124    def menu_list_view(request: Request):
125        user: AbstractUser = request.user
126        auth = request.auth
127        print(user, auth)
128        # print(dir(user)) # .is_superuser
129        menu_list = {
130            'data': []
131        }
132        if user.is_superuser: # 用户管理必须管理员
133            p1 = PermissionItem(101, '用户管理')
134            p1.append(PermissionItem(102, '用户列表', '/users/'))
135            p1.append(PermissionItem(103, '角色列表', '/users/roles/'))
136            p1.append(PermissionItem(104, '权限列表', '/users/perms/'))
137            menu_list['data'].append(p1)
138
139        # 权限测试
140        if user.has_perm('user.view_userprofile'):
141            p9 = PermissionItem(901, '管理')
142            p9.append(PermissionItem(902, 'x列表', '/users/'))
143            p9.append(PermissionItem(903, 'y列表', '/users/roles/'))
144            p9.append(PermissionItem(904, 'z列表', '/users/perms/'))
145            menu_list['data'].append(p9)
146
147        # 普通用户测试
148        p10 = PermissionItem(1001, '测试项')
149        p10.append(PermissionItem(1002, '测试用', '/test'))
150        menu_list['data'].append(p10)
151        return Response(menu_list)

```

---

使用权限，灵活控制各模块权限

