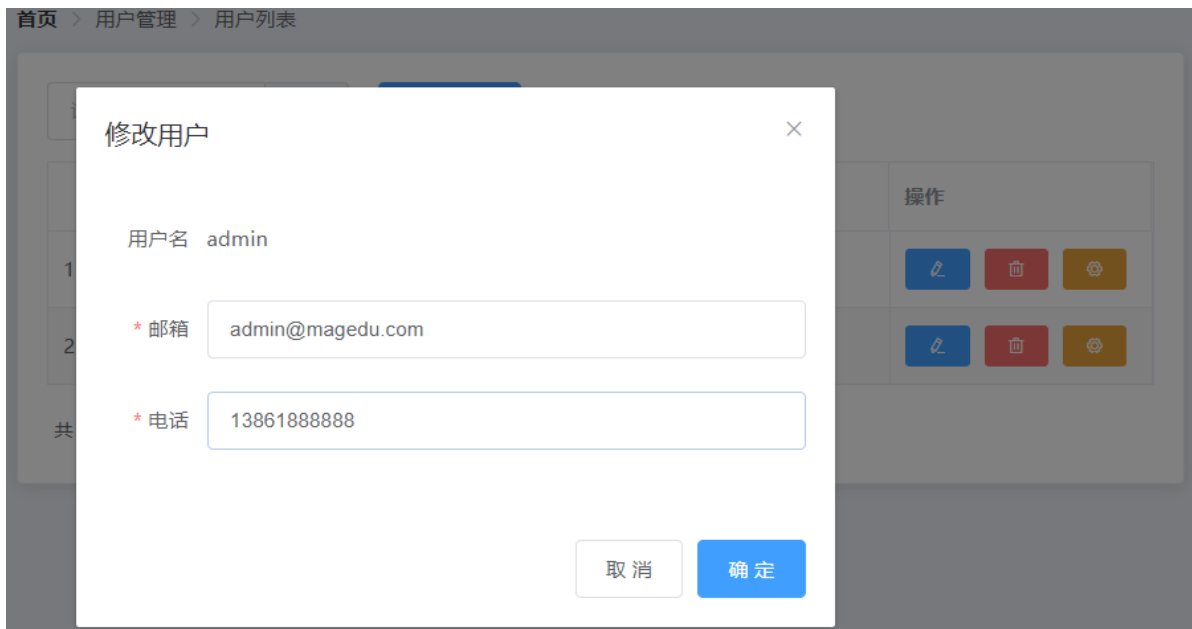


用户功能实现

功能有：

- 用户列表、分页
- 添加用户、用户修改、用户删除
- 激活、禁用用户
- 用户搜索
- 角色配置



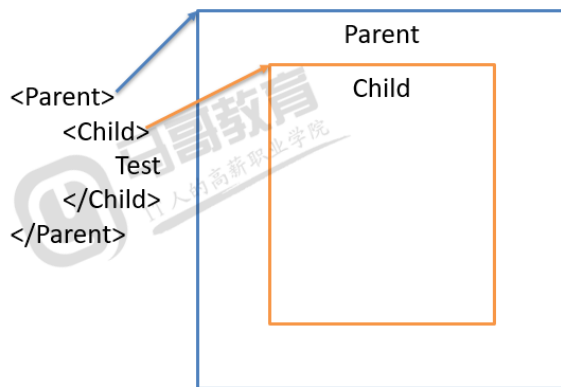


Vue插槽

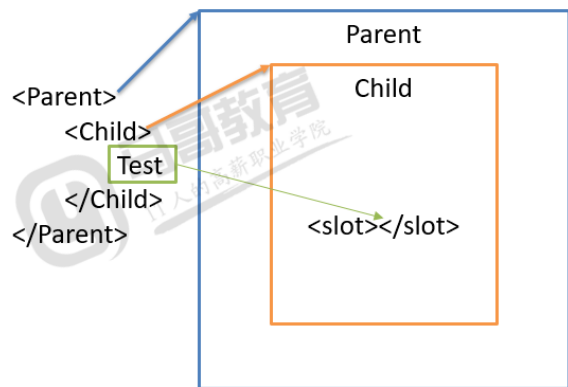
参考 <https://cn.vuejs.org/v2/guide/components-slots.html>

插槽

插槽，指的是在子组件中挖个空（slot插槽），在父组件中使用子组件标签内的内容，来填充子组件的空。



不能显示Test，因为没有插槽



插槽处替换为Test

注意：以上是原理图，不能和Vue的实际代码完全对应

插槽测试

新建T1.vue和T2.vue，T1是父组件，T2是子组件。

```

1  <template>
2    <div>
3      父组件T1内容
4      <hr />
5      <T2>父组件中在子组件插入的内容c</T2>
6    </div>
7  </template>
8
9  <script>
10 import T2 from './T2.vue'
11 export default {
12   components: {
13     T2
14   }
15 }
16 </script>
17
18 <style>
19 </style>

```

```

1  <template>
2    <div>子组件T2内容</div>
3  </template>
4
5  <script>
6  export default {}
7  </script>
8
9  <style>
10 </style>

```

父组件T1内容

子组件T2内容

发现内容c并没有显示出来。这就需要在子组件中使用插槽来接收。

修改T2增加插槽，代码如下

```

1 <template>
2   <div>
3     子组件T2内容
4     <div style="border: 1px solid #000"><slot></slot></div>
5   </div>
6 </template>
7
8 <script>
9   export default {}
10 </script>
11
12 <style>
13 </style>

```

父组件T1内容

子组件T2内容

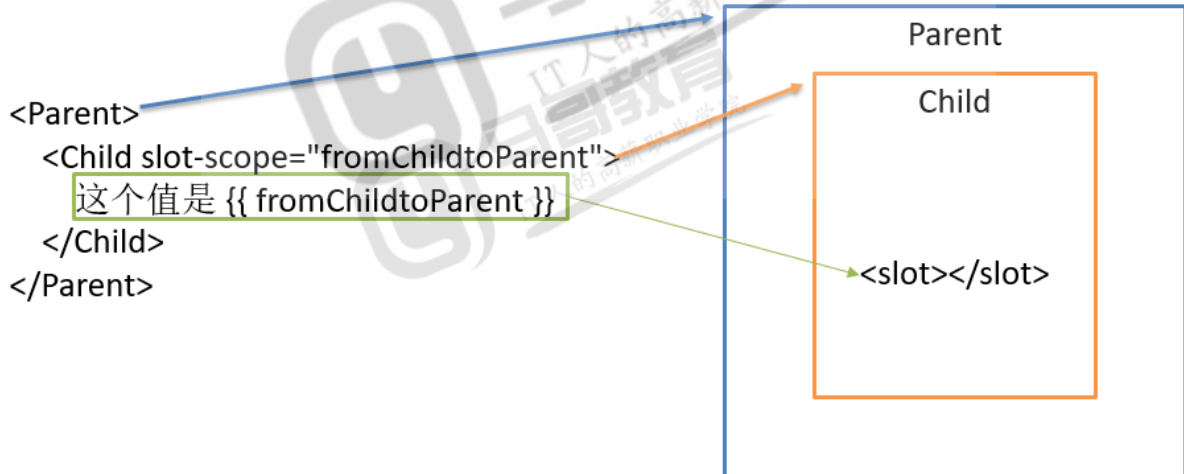
父组件中在子组件插入的内容c

插槽：父组件通过子组件标签为子组件传递内容，但是子组件内部要使用插槽接收。

作用域插槽

插槽可以看做数据从父组件流向子组件，但有时父组件需要使用子组件中的数据，怎么办？

作用域插槽Scoped Slot，在父组件中能访问到子组件的数据。



插槽处被替换为绿框的内容，不过fromChildtoParent来自于子组件的data的数据

作用域插槽：父组件中为子组件传递的数据不在父组件中，数据在子组件中，只能通过作用域插槽从子组件向父组件传过来这些数据，最后这些数据可以在父组件中使用，但是显示在子组件的插槽中。

新语法

slot-scope="fromChildtoParent" 即将过期，非命名插槽，可以使用新语法

#default="fromChildtoParent"

作用域插槽测试

修改子组件T2如下

```
1 <template>
2   <div>
3     子组件T2内容
4     <div style="border: 1px solid #000">
5       通过插槽bind数据，这样父组件就可以使用这个数据了，user是属性名<br /><br />
6       <slot :user="u"></slot>
7     </div>
8   </div>
9 </template>
10
11 <script>
12 export default {
13   data() {
14     return {
15       u: { name: 'tom', age: 20 }
16     }
17   }
18 }
19 </script>
20
21 <style>
22 </style>
```

修改父组件T1如下，如何使用子组件中的user？

```
1 <template>
2   <div>
3     父组件T1内容
4     <hr />
5     <T2>
6       <template v-slot:default="xyz"> 从子组件中传来的数据是：{{ xyz.user.name }} </template>
7     </T2>
8   </div>
9 </template>
10
11 <script>
12 import T2 from './T2.vue'
13 export default {
14   components: {
15     T2
16   }
17 }
18 </script>
19
20 <style>
21 </style>
```

可以继续简化为

```
1 <T2 #default="xyz"> 从子组件中传来的数据是：{{ xyz.user.age }} </T2>
```

顺便可以解构

```
1 <T2 #default="{ user }"> 从子组件中传来的数据是: {{ user.name }}, {{ user.age }}
  </T2>
```

激活功能

使用switch开关组件表示是否激活状态。

src/plugins/element.js

```
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
7 } from 'element-ui'
8
9 Vue.use(Switch)
```

采用下面的方式，不能把数据注入给switch组件

```
1 <el-table-column prop="is_active" label="激活">
2   <el-switch v-model="is_active"> </el-switch>
3 </el-table-column>
```

参考 <https://element.eleme.cn/#/zh-CN/component/table#zi-ding-yi-lie-mo-ban>

通过作用域插槽 `Scoped slot` 可以获取到 `row`, `column`, `$index` 和 `store` (table 内部的状态管理) 的数据

从这个例子可以看出，`scope`是来自表格子组件的数据，我们在父组件中想用这个值，`template`的内容最终插入到表格的列中的插槽中。`scope.row`就是取当前行数据对象。

```
1 <el-table :data="userList" border style="width: 100%">
2   <el-table-column type="index" label="序号"> </el-table-column>
3   <el-table-column prop="username" label="登录名"> </el-table-column>
4   <el-table-column label="激活">
5     <template #default="scope">
6       <el-switch v-model="scope.row.is_active"></el-switch>
7     </template>
8   </el-table-column>
9   <el-table-column prop="is_superuser" label="管理员"> </el-table-
column>
10   <el-table-column prop="phone" label="电话"> </el-table-column>
11   <el-table-column label="操作">
12     <el-button type="success" icon="el-icon-edit" size="mini"></el-
button>
13     <el-button type="danger" icon="el-icon-delete" size="mini"></el-
button>
14   </el-table-column>
```

上面代码顺便增加了2个按钮编辑和删除。`<el-table-column type="index" label="序号"> </el-table-column>` 为表格增加序号。

Switch组件会发生change事件，改变值，就需要去改变数据库中某个用户的is_active的值

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="input3">
13              <el-button slot="append" icon="el-icon-search"></el-button>
14            </el-input>
15          </div>
16        </el-col>
17        <el-col :span="12">
18          <el-button type="primary" @click="addUserDialogVisible = true">增
19          加用户</el-button>
20        </el-col>
21      </el-row>
22      <el-table :data="userList" border style="width: 100%">
23        <el-table-column type="index" label="序号"> </el-table-column>
24        <el-table-column prop="username" label="用户名"> </el-table-column>
25        <el-table-column prop="is_active" label="激活">
26          <template #default="{ row }">
27            <el-switch v-model="row.is_active"
28              @change="handleIsActiveChange(row)"> </el-switch>
29          </template>
30        </el-table-column>
31        <el-table-column prop="is_superuser" label="管理员"> </el-table-
32        column>
33        <el-table-column prop="phone" label="电话"> </el-table-column>
34        <el-table-column label="操作">
35          <el-button type="success" icon="el-icon-edit" size="mini"></el-
36          button>
37          <el-button type="danger" icon="el-icon-delete" size="mini"></el-
38          button>
39        </el-table-column>
40      </el-table>
41      <el-pagination
42        @current-change="handleCurrentChange"
43        :current-page="pagination.page"
44        :page-size="pagination.size"
45        layout="total, prev, pager, next, jumper"
46        :total="pagination.total"
47      >

```

```

43     </el-pagination>
44   </el-card>
45   <!-- 添加用户对话框 -->
46   <el-dialog title="增加" :visible.sync="addUserDialogVisible"
@close="resetForm('addUser')">
47     <el-form :model="addUserForm" :rules="addUserRules" ref="addUser"
label-width="100px">
48       <el-form-item label="用户名" prop="username">
49         <el-input v-model="addUserForm.username"></el-input>
50       </el-form-item>
51       <el-form-item label="密码" prop="password">
52         <el-input type="password" v-model="addUserForm.password"></el-
input>
53       </el-form-item>
54       <el-form-item label="确认密码" prop="checkPass">
55         <el-input type="password" v-model="addUserForm.checkPass"></el-
input>
56       </el-form-item>
57       <el-form-item label="电话" prop="phone">
58         <el-input v-model="addUserForm.phone"></el-input>
59       </el-form-item>
60       <el-form-item label="姓" prop="last_name">
61         <el-input v-model="addUserForm.last_name"></el-input>
62       </el-form-item>
63       <el-form-item label="名" prop="first_name">
64         <el-input v-model="addUserForm.first_name"></el-input>
65       </el-form-item>
66       <el-form-item label="邮箱" prop="email">
67         <el-input v-model="addUserForm.email"></el-input>
68       </el-form-item>
69     </el-form>
70     <span slot="footer" class="dialog-footer">
71       <el-button @click="addUserDialogVisible = false">取 消</el-button>
72       <el-button type="primary" @click="addUser">确 定</el-button>
73     </span>
74   </el-dialog>
75 </div>
76 </template>
77
78 <script>
79 export default {
80   created() {
81     this.getUserList()
82   },
83   data() {
84     const validatePass = (rule, value, callback) => {
85       if (value !== this.addUserForm.password) {
86         callback(new Error('两次输入密码不一致!'))
87       } else {
88         callback()
89       }
90     }
91     return {
92       input3: '',
93       addUserDialogVisible: false,
94       addUserForm: {
95         username: '',
96         password: '',

```



```

97     phone: '',
98     first_name: '',
99     last_name: '',
100    email: ''
101  },
102  addUserRules: {
103    username: [
104      { required: true, message: '请输入登录用户名', trigger: 'blur' },
105      { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
106    ],
107    password: [
108      { required: true, message: '请输入密码', trigger: 'blur' },
109      { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
110    ],
111    checkPass: [
112      { required: true, message: '请输入密码', trigger: 'blur' },
113      { validator: validatePass, trigger: 'blur' }
114    ]
115  },
116  userList: [],
117  pagination: { page: 1, size: 20, total: 0 }
118  }
119 },
120 methods: {
121   resetForm(name) {
122     console.log('~~~~~')
123     this.$refs[name].resetFields()
124   },
125   addUser() {
126     const name = 'addUser'
127     this.$refs[name].validate(async (valid) => {
128       if (valid) {
129         const { data: response } = await this.$http.post('users/',
this.addUserForm)
130         if (response.code) {
131           return this.$message.error(response.message)
132         }
133         this.addUserDialogVisible = false
134         this.resetForm(name)
135         this.getUserList()
136       }
137     })
138   },
139   async getUserList(page = 1) {
140     if (!page) {
141       page = 1
142     }
143     const { data: response } = await this.$http.get(`users/?
page=${page}`)
144     if (response.code) {
145       return this.$message.error(response.message)
146     }
147     console.log(response)
148     this.userList = response.results
149     this.pagination = response.pagination
150   },

```

```

151     handleCurrentChange(val) {
152         console.log(`当前页: ${val}`)
153         this.getUserList(val)
154     },
155     async handleIsActiveChange(row) {
156         // 激活按钮变化, 去后台部分更新字段
157         await this.$http.patch(`users/${row.id}/`, { is_active: row.is_active
158     })
159     }
160 }
161 </script>
162
163 <style lang="less" scoped></style>

```

用户搜索功能

仔细思考一下, 这个功能也是用的list方法, 发送GET请求到/users/, 只不过需要带上参数, 也就是查询字符串。

阅读Axios文档, GET请求可以使用 `axios.get(url[, config])`, 而config可以用下面的方式传参。这种方式本质上还是查询字符串方式。

```

1  axios.get('users/', {
2      params: {
3          page: 4
4      }
5  })

```

所以, 可以将getUserList(page), 修改如下

```

1  async getUserList(page = 1) {
2      if (!page) {
3          page = 1
4      }
5      const { data: response } = await this.$http.get('users/', {
6          params: {
7              page,
8              username: this.search
9          }
10     })
11     if (response.code) {
12         return this.$message.error(response.message)
13     }
14     this.userList = response.results
15     this.pagination = response.pagination
16 }

```

下面为了聚焦在改动的代码上, 省略一些没改过的代码

```

1  <template>
2      <div>
3          <el-card class="box-card">
4              <el-row :gutter="20">

```

```

5         <el-col :span="12">
6             <el-input placeholder="请输入内容" v-model="keywords">
7                 <el-button slot="append" icon="el-icon-search"
@click="getUserList(1)"></el-button>
8             </el-input>
9         </el-col>
10        <el-col :span="12">
11            <el-button type="primary" @click="addUserDialogVisible = true">增加
用户</el-button>
12        </el-col>
13    </el-row>
14 </el-card>
15 </div>
16 </template>
17
18 <script>
19 export default {
20     created() {
21         this.getUserList()
22     },
23     data() {
24         return {
25             keywords: '',
26             // 用户列表表格
27             userList: [],
28             pagination: { page: 1, size: 20, total: 0 }
29         }
30     },
31     methods: {
32         async getUserList(page = 1) {
33             if (!page) {
34                 page = 1
35             }
36             const { data: response } = await this.$http.get('users/', {
37                 params: {
38                     page,
39                     username: this.keywords
40                 }
41             })
42             if (response.code) {
43                 return this.$message.error(response.message)
44             }
45             this.userList = response.results
46             this.pagination = response.pagination
47         }
48     }
49 }
50 </script>

```

发起请求时，username是发到后台了，但是没有用，如何解决，这个就必须要覆盖GenericAPIView的get_queryset()了。

```

1 from rest_framework.viewsets import ModelViewSet
2 from django.contrib.auth import get_user_model
3 from .serializers import UserSerializer
4
5 class UserViewSet(ModelViewSet):

```

```

6     queryset = get_user_model().objects.all()
7     serializer_class = UserSerializer
8     # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
9
10    def get_queryset(self):
11        queryset = super().get_queryset() # 调用父类的处理
12        username = self.request.query_params.get('username', None)
13        if username:
14            queryset = queryset.filter(username__icontains=username)
15        return queryset

```

由于这个用到了列表页的list方法，查看ListModelMixin源码中，list方法返回列表前调用了filter_queryset，就是用来过滤的，覆盖filter_queryset也行。

这是一个通过的功能，以后免不了要经常查询，但是，分页功能我们知道了，不是什么参数，服务器端都允许的，这需要配置。

通用查询

参考 <https://www.django-rest-framework.org/api-guide/filtering/#djangofilterbackend>

```

1 pip install django-filter

```

```

1 INSTALLED_APPS = [
2     ...
3     'django_filters',
4     ...
5 ]

```

```

1 REST_FRAMEWORK = {
2     'DEFAULT_FILTER_BACKENDS':
3     ['django_filters.rest_framework.DjangoFilterBackend']
4 }

```

```

1 from rest_framework.permissions import IsAuthenticated, IsAdminUser
2 from rest_framework.viewsets import ModelViewSet
3 from django.contrib.auth import get_user_model
4 from .models import UserProfile
5 from .serializers import UserSerializer
6 from django_filters.rest_framework import DjangoFilterBackend
7
8 class UserViewSet(ModelViewSet):
9     queryset = get_user_model().objects.all()
10    serializer_class = UserSerializer
11    # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
12    filter_backends = [DjangoFilterBackend] # 指定filter
13    filterset_fields = ['username']

```

请求为 `http://localhost:8080/api/v1/users/?page=1&username=admin`

搜索SQL语句条件为 `FROM auth_user WHERE auth_user.username = 'admin'`

这是等值条件，要求查询参数名必须为filterset_fields中指定的名称。

实现模糊搜索，代码如下

```

1 from rest_framework import filters
2
3 class UserViewSet(ModelViewSet):
4     queryset = get_user_model().objects.all()
5     serializer_class = UserSerializer
6     # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
7     filter_backends = [filters.SearchFilter] # 指定filter
8     # filterset_fields = ['username']
9     search_fields = ['username']

```

请求为 `http://localhost:8080/api/v1/users/?page=1&search=ay`，注意是search=

搜索SQL语句条件为 `FROM auth_user WHERE auth_user.username LIKE '%ay%'`

`search_fields = ['username']` 中如果写多个字段将是或查询。

修改用户

前台实现，仿照增加用户，部分代码如下

```

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-table :data="userList" border style="width: 100%">
10        <el-table-column type="index" label="序号"> </el-table-column>
11        <el-table-column prop="username" label="用户名"> </el-table-column>
12        <el-table-column prop="is_active" label="激活">
13          <template #default="{ row }">
14            <el-switch v-model="row.is_active"
15              @change="handleIsActiveChange(row)"> </el-switch>
16          </template>
17        </el-table-column>
18        <el-table-column prop="is_superuser" label="管理员"> </el-table-
19column>
20        <el-table-column prop="phone" label="电话"> </el-table-column>
21        <el-table-column label="操作">
22          <template #default="{ row }">
23            <el-button @click="handleEditUser(row)" type="success"
24              icon="el-icon-edit" size="mini"></el-button>
25            <el-button type="danger" icon="el-icon-delete" size="mini">
26              </el-button>
27          </template>
28        </el-table-column>
29      </el-table>
30      <el-pagination
31        @current-change="handleCurrentChange"
32        :current-page="pagination.page"
33        :page-size="pagination.size"
34        layout="total, prev, pager, next, jumper"
35        :total="pagination.total"

```

```

32     >
33     </el-pagination>
34 </el-card>
35 <!-- 修改用户对话框 -->
36 <el-dialog title="修改" :visible.sync="editUserDialogVisible"
@close="resetForm('editUser')">
37     <el-form :model="editUserForm" :rules="editUserRules" ref="editUser"
label-width="100px">
38         <el-form-item label="用户名" prop="username">{{
editUserForm.username }}</el-form-item>
39         <el-form-item label="电话" prop="phone">
40             <el-input v-model="editUserForm.phone"></el-input>
41         </el-form-item>
42         <el-form-item label="姓" prop="last_name">
43             <el-input v-model="editUserForm.last_name"></el-input>
44         </el-form-item>
45         <el-form-item label="名" prop="first_name">
46             <el-input v-model="editUserForm.first_name"></el-input>
47         </el-form-item>
48         <el-form-item label="邮箱" prop="email">
49             <el-input v-model="editUserForm.email"></el-input>
50         </el-form-item>
51     </el-form>
52     <span slot="footer" class="dialog-footer">
53         <el-button @click="editUserDialogVisible = false">取 消</el-button>
54         <el-button type="primary" @click="editUser">确 定</el-button>
55     </span>
56 </el-dialog>
57 </div>
58 </template>
59
60 <script>
61 export default {
62     data() {
63         return {
64             search: '',
65             // 修改用户对话框
66             editUserDialogVisible: false,
67             editUserForm: {
68                 username: '',
69                 phone: '',
70                 first_name: '',
71                 last_name: '',
72                 email: ''
73             },
74             editUserRules: {}
75         }
76     },
77     methods: {
78         resetForm(name) {
79             console.log('~~~~~')
80             this.$refs[name].resetFields()
81         },
82         handleEditUser(row) {
83             this.editUserForm = row
84             this.editUserDialogVisible = true
85         },
86         editUser() {

```

```

87     const { id } = this.editUserForm
88     const name = 'editUser'
89     this.$refs[name].validate(async (valid) => {
90         if (valid) {
91             const { data: response } = await this.$http.patch(`users/${id}/`,
this.editUserForm)
92             if (response.code) {
93                 return this.$message.error(response.message)
94             }
95             this.editUserDialogVisible = false
96             this.resetForm(name)
97             this.getUserList(this.pagination.page) // 刷新用户列表
98         }
99     })
100 }
101 }
102 }
103 </script>
104
105 <style lang="less" scoped></style>

```

后台如何做到，修改用户时，有id，用的是详情页，但有人故意构造提交的数据中包含username，如果后端也更新了，就修改了用户名，按要求不允许修改用户名，怎么办？还是用PATCH，部分更新（本质上就是先查后部分更新），在后端要剔除掉request数据中的username，这需要覆盖partial_update(self, request, *args, **kwargs)。

```

1  from rest_framework.permissions import IsAuthenticated, IsAdminUser
2  from rest_framework.viewsets import ModelViewSet
3  from django.contrib.auth import get_user_model
4  from .models import UserProfile
5  from .serializers import UserSerializer
6  from django_filters.rest_framework import DjangoFilterBackend
7  from rest_framework import filters
8
9  class UserViewSet(ModelViewSet):
10     queryset = get_user_model().objects.all()
11     serializer_class = UserSerializer
12     # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
13     filter_backends = [filters.SearchFilter] # 指定filter
14     # filterset_fields = ['username']
15     search_fields = ['username']
16
17     # 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
18     def partial_update(self, request, *args, **kwargs):
19         request.data.pop('username', None) # 剔除不要更新的字段
20         request.data.pop('id', None)
21         request.data.pop('password', None)
22         return super().partial_update(request, *args, **kwargs)

```

确认消息框和提示

确认框参考 <https://element.eleme.cn/#/zh-CN/component/message-box#que-ren-xiao-xi>

提示文字参考 <https://element.eleme.cn/#/zh-CN/component/tooltip>

src/plugins/element.js

```
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 } from 'element-ui'
8
9 Vue.use(Tooltip)
10
11 // 全局导入
12 Vue.prototype.$message = Message
13 Vue.prototype.$msgbox = MessageBox
```

```
1 <el-table :data="userList" border style="width: 100%">
2   <el-table-column type="index" label="序号"> </el-table-column>
3   <el-table-column prop="username" label="用户名"> </el-table-column>
4   <el-table-column prop="is_active" label="激活">
5     <template #default="{ row }">
6       <el-switch v-model="row.is_active"
7         @change="handleIsActiveChange(row)"> </el-switch>
8     </template>
9   </el-table-column>
10   <el-table-column prop="is_superuser" label="管理员"> </el-table-
11 column>
12   <el-table-column prop="phone" label="电话"> </el-table-column>
13   <el-table-column label="操作">
14     <template #default="{ row }">
15       <el-tooltip v-if="row.id !== 1" content="修改" effect="light">
16         <el-button @click="handleEditUser(row)" type="success"
17           icon="el-icon-edit" size="mini"></el-button>
18       </el-tooltip>
19       <el-tooltip v-if="row.id !== 1" content="删除" effect="light">
20         <el-button type="danger" icon="el-icon-delete" size="mini">
21       </el-button>
22     </template>
23   </el-table-column>
24 </el-table>
```

删除用户

前端对id为1的用户，使用v-if禁止修改、删除，使用disabled禁止激活。注意，前端的禁止行为只是一种样子，依然可以绕过它，后端禁止才是关键。

```
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="修改" effect="light">
38              <el-button @click="handleEditUser(row)" type="success"
39                icon="el-icon-edit" size="mini"></el-button>
40            </el-tooltip>
41            <el-tooltip v-if="row.id !== 1" content="删除" effect="light">
42              <el-button @click="handleDelUser(row.id)" type="danger"
43                icon="el-icon-delete" size="mini"></el-button>
44            </el-tooltip>
45          </template>
46        </el-table-column>
47      </el-table>
48      <el-pagination
49        @current-change="handleCurrentChange"
50        :current-page="pagination.page"
51        :page-size="pagination.size"
52        layout="total, prev, pager, next, jumper"
53        :total="pagination.total"
54      >
55    </el-pagination>
56  </el-card>
57  <!-- 修改用户对话框 -->
58  <el-dialog title="修改" :visible.sync="editUserDialogVisible"
59    @close="resetForm('editUser')">

```

```

53     <el-form :model="editUserForm" :rules="editUserRules" ref="editUser"
label-width="100px">
54     <el-form-item label="用户名" prop="username">{{
editUserForm.username }}</el-form-item>
55     <el-form-item label="电话" prop="phone">
56     <el-input v-model="editUserForm.phone"></el-input>
57     </el-form-item>
58     <el-form-item label="姓" prop="last_name">
59     <el-input v-model="editUserForm.last_name"></el-input>
60     </el-form-item>
61     <el-form-item label="名" prop="first_name">
62     <el-input v-model="editUserForm.first_name"></el-input>
63     </el-form-item>
64     <el-form-item label="邮箱" prop="email">
65     <el-input v-model="editUserForm.email"></el-input>
66     </el-form-item>
67   </el-form>
68   <span slot="footer" class="dialog-footer">
69     <el-button @click="editUserDialogVisible = false">取 消</el-button>
70     <el-button type="primary" @click="editUser">确 定</el-button>
71   </span>
72 </el-dialog>
73 <!-- 添加用户对话框 -->
74 <el-dialog title="增加" :visible.sync="addUserDialogVisible"
@close="resetForm('addUser')">
75   <el-form :model="addUserForm" :rules="addUserRules" ref="addUser"
label-width="100px">
76     <el-form-item label="用户名" prop="username">
77     <el-input v-model="addUserForm.username"></el-input>
78     </el-form-item>
79     <el-form-item label="密码" prop="password">
80     <el-input type="password" v-model="addUserForm.password"></el-
input>
81     </el-form-item>
82     <el-form-item label="确认密码" prop="checkPass">
83     <el-input type="password" v-model="addUserForm.checkPass"></el-
input>
84     </el-form-item>
85     <el-form-item label="电话" prop="phone">
86     <el-input v-model="addUserForm.phone"></el-input>
87     </el-form-item>
88     <el-form-item label="姓" prop="last_name">
89     <el-input v-model="addUserForm.last_name"></el-input>
90     </el-form-item>
91     <el-form-item label="名" prop="first_name">
92     <el-input v-model="addUserForm.first_name"></el-input>
93     </el-form-item>
94     <el-form-item label="邮箱" prop="email">
95     <el-input v-model="addUserForm.email"></el-input>
96     </el-form-item>
97   </el-form>
98   <span slot="footer" class="dialog-footer">
99     <el-button @click="addUserDialogVisible = false">取 消</el-button>
100    <el-button type="primary" @click="addUser">确 定</el-button>
101  </span>
102 </el-dialog>
103 </div>
104 </template>

```

```

105
106 <script>
107 export default {
108   created() {
109     this.getUserList()
110   },
111   data() {
112     const validatePass = (rule, value, callback) => {
113       if (value !== this.addUserForm.password) {
114         callback(new Error('两次输入密码不一致!'))
115       } else {
116         callback()
117       }
118     }
119     return {
120       search: '',
121       addUserDialogvisible: false,
122       addUserForm: {
123         username: '',
124         password: '',
125         phone: '',
126         first_name: '',
127         last_name: '',
128         email: ''
129       },
130       addUserRules: {
131         username: [
132           { required: true, message: '请输入登录用户名', trigger: 'blur' },
133           { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
134             'blur' }
135         ],
136         password: [
137           { required: true, message: '请输入密码', trigger: 'blur' },
138           { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
139             'blur' }
140         ],
141         checkPass: [
142           { required: true, message: '请输入密码', trigger: 'blur' },
143           { validator: validatePass, trigger: 'blur' }
144         ]
145       },
146       userList: [],
147       pagination: { page: 1, size: 20, total: 0 },
148       // 修改用户对话框
149       editUserDialogvisible: false,
150       editUserForm: {
151         username: '',
152         phone: '',
153         first_name: '',
154         last_name: '',
155         email: ''
156       },
157       editUserRules: {}
158     },
159     methods: {
160       resetForm(name) {
161         console.log('~~~~~')

```

```

161     this.$refs[name].resetFields()
162   },
163   addUser() {
164     const name = 'addUser'
165     this.$refs[name].validate(async (valid) => {
166       if (valid) {
167         const { data: response } = await this.$http.post('users/',
this.addUserForm)
168         if (response.code) {
169           return this.$message.error(response.message)
170         }
171         this.addUserDialogvisible = false
172         this.resetForm(name)
173         this.getUserList()
174       }
175     })
176   },
177   async getUserList(page = 1) {
178     if (!page) {
179       page = 1
180     }
181     const { data: response } = await this.$http.get('users/', {
182       params: {
183         page,
184         search: this.search
185       }
186     })
187     if (response.code) {
188       return this.$message.error(response.message)
189     }
190     this.userList = response.results
191     this.pagination = response.pagination
192   },
193   handleCurrentChange(val) {
194     console.log(`当前页: ${val}`)
195     this.getUserList(val)
196   },
197   async handleIsActiveChange(row) {
198     // 激活按钮变化，去后台部分更新字段
199     await this.$http.patch(`users/${row.id}/`, { is_active: row.is_active
200   })
201   },
202   handleEditUser(row) {
203     this.editUserForm = row
204     this.editUserDialogvisible = true
205   },
206   editUser() {
207     const { id } = this.editUserForm
208     const name = 'editUser'
209     this.$refs[name].validate(async (valid) => {
210       if (valid) {
211         const { data: response } = await this.$http.patch(`users/${id}/`,
this.editUserForm)
212         if (response.code) {
213           return this.$message.error(response.message)
214         }
215         this.editUserDialogvisible = false
216         this.resetForm(name)

```

```

216         this.getUserList(this.pagination.page) // 刷新用户列表
217     }
218 })
219 },
220 // 删除用户
221 handleDelUser(id) {
222     this.$msgbox
223         .confirm('删除该用户，是否继续?', '提示', {
224             confirmButtonText: '确定',
225             cancelButtonText: '取消',
226             type: 'danger',
227             center: true
228         })
229         .then(async () => {
230             const { data: response } = await
231             this.$http.delete(`users/${id}/`)
232             if (response.code) {
233                 return this.$message.error(response.message)
234             }
235             this.getUserList()
236         })
237         .catch(() => {})
238     }
239 }
240 </script>
241
242 <style lang="less" scoped></style>

```

顺便考虑一下id为1，就是管理员，后端禁止修改和删除，这个需要修改什么方法？

```

1  from rest_framework.permissions import IsAuthenticated, IsAdminUser
2  from rest_framework.viewsets import ModelViewSet
3  from django.contrib.auth import get_user_model
4  from .models import UserProfile
5  from .serializers import UserSerializer
6  from django_filters.rest_framework import DjangoFilterBackend
7  from rest_framework import filters
8  from django.http.response import Http404
9
10 class UserViewSet(ModelViewSet):
11     queryset = get_user_model().objects.all()
12     serializer_class = UserSerializer
13     # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
14     filter_backends = [filters.SearchFilter] # 指定filter
15     # filterset_fields = ['username']
16     search_fields = ['username']
17
18     # 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
19     def partial_update(self, request, *args, **kwargs):
20         request.data.pop('username', None) # 剔除不要更新的字段
21         request.data.pop('id', None)
22         request.data.pop('password', None)
23         return super().partial_update(request, *args, **kwargs)
24

```

```

25     def get_object(self):
26         if self.request.method.lower() != 'get':
27             pk = self.kwargs.get('pk')
28             if pk == 1 or pk == '1':
29                 raise Http404
30             return super().get_object()

```

用户信息菜单

下拉菜单参考 <https://element.eleme.cn/#/zh-CN/component/dropdown>

src/plugins/element.js

```

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
8  } from 'element-ui'
9
10 Vue.use(Dropdown)
11 Vue.use(DropdownMenu)
12 Vue.use(DropdownItem)

```

src/components/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                 <el-dropdown @command="handleCommand">
12                     <span class="el-dropdown-link">{{ user.username }}<i class="el-
icon-arrow-down el-icon--right"></i></span>
13                     <el-dropdown-menu slot="dropdown">
14                         <el-dropdown-item command="chpwd">修改密码</el-dropdown-item>
15                         <el-dropdown-item command="logout">退出</el-dropdown-item>
16                     </el-dropdown-menu>
17                 </el-dropdown>
18             </div>
19         </el-header>
20         <el-container>
21             <el-aside :width="isCollapsed ? '64px' : '200px'">
22                 <el-menu
23                     background-color="#123"
24                     text-color="#fff"
25                     active-text-color="#ffd04b"

```

```

26         :collapse="isCollapsed"
27         :collapse-transition="true"
28         :router="true"
29     >
30         <el-submenu v-for="item in menulist" :index="item.id + '"
:key="item.id">
31             <template slot="title">
32                 <i class="el-icon-menu"></i><span slot="title">{{
item.itemName }}</span>
33             </template>
34             <el-menu-item v-for="subItem in item.children"
:index="subItem.path" :key="subItem.id">
35                 <i class="el-icon-menu"></i><span slot="title">{{
subItem.itemName }}</span>
36             </el-menu-item>
37         </el-submenu>
38         <el-menu-item index="/welcome">
39             <i class="el-icon-setting"></i>
40             <span slot="title">导航四</span>
41         </el-menu-item>
42         <el-menu-item index="/t1">
43             <i class="el-icon-setting"></i>
44             <span slot="title">t1</span>
45         </el-menu-item>
46     </el-menu>
47 </el-aside>
48 <el-main>
49     <router-view></router-view>
50 </el-main>
51 </el-container>
52 </el-container>
53 </template>

```

用户信息

登录成功，返回JSON的Token，Header部分显示用户信息，比如用户id、用户名等。

src/components/Home.vue

```

1  <script>
2  export default {
3      created() {
4          this.getMenuList() // vue组件创建时查菜单数据
5          this.getUserInfo()
6      },
7      data() {
8          return {
9              menulist: [], // 菜单数据
10             isCollapsed: false,
11             user: {}, // 用户信息,
12             chpwdDialogVisible: false // 修改密码对话框
13         }
14     },
15     methods: {
16         async getMenuList() {
17             const { data: response } = await this.$http.get('users/menulist/')
18             if (response.code) {

```

```

19         return this.$message.error(response.message)
20     }
21     this.menulist = response.data
22 },
23 logout() {
24     window.localStorage.removeItem('token')
25     this.$router.push('/login')
26 },
27 toggleMenu() {
28     this.isCollapsed = !this.isCollapsed
29 },
30 handleCommand(command) {
31     if (command === 'logout') {
32         this.logout()
33     } else if (command === 'chpwd') {
34         this.chpwdDialogVisible = true
35     }
36 },
37 async getUserInfo() {
38     // 带着token发起请求, 后台查询用户数据
39     const { data: response } = await this.$http.get('users/whoami/')
40     if (response.code) {
41         return this.$message.error(response.message)
42     }
43     this.user = response.user
44 },
45 changePassword() {}
46 }
47 }
48 </script>

```

user/views.py

```

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
5  from django.contrib.auth import get_user_model
6  from .models import UserProfile
7  from .serializers import UserSerializer
8  from django_filters.rest_framework import DjangoFilterBackend
9  from rest_framework import filters
10 from django.http.response import Http404
11
12 class UserViewSet(ModelViewSet):
13     queryset = get_user_model().objects.all()
14     serializer_class = UserSerializer
15     # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
16     filter_backends = [filters.SearchFilter] # 指定filter
17     # filterset_fields = ['username']
18     search_fields = ['username']
19
20     # 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
21     def partial_update(self, request, *args, **kwargs):
22         request.data.pop('username', None) # 剔除不要更新的字段

```



```

23     request.data.pop('id', None)
24     request.data.pop('password', None)
25     return super().partial_update(request, *args, **kwargs)
26
27     def get_object(self):
28         if self.request.method.lower() != 'get':
29             pk = self.kwargs.get('pk')
30             if pk == 1 or pk == '1':
31                 raise Http404
32             return super().get_object()
33
34     @action(['GET'], detail=False, url_path='whoami')
35     def whoami(self, request): # detail=False, 非详情页
36         print(request.user)
37         return Response({
38             'user': {
39                 'id': request.user.id,
40                 'username': request.user.username
41             }
42         })

```

去看看自动生成的路由多了什么？

密码修改

使用对话框，提供当前密码和新密码。

src/components/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-dropdown @command="handleCommand">
11           <span class="el-dropdown-link">{{ user.username }}<i class="el-
icon-arrow-down el-icon--right"></i></span>
12           <el-dropdown-menu slot="dropdown">
13             <el-dropdown-item command="chpwd">修改密码</el-dropdown-item>
14             <el-dropdown-item command="logout">退出</el-dropdown-item>
15           </el-dropdown-menu>
16         </el-dropdown>
17       </div>
18       <!-- 修改密码对话框 -->
19       <el-dialog title="修改密码" :visible.sync="chpwdDialogVisible"
@close="resetForm('chpwd')">
20         <el-form :model="chpwdForm" :rules="chpwdRules" ref="chpwd" label-
width="100px">
21           <el-form-item label="用户名">{{ user.username }}</el-form-item>
22           <el-form-item label="旧密码" prop="oldPassword">

```

```

23         <el-input type="password" v-model="chpwdForm.oldPassword"></el-
input>
24     </el-form-item>
25     <el-form-item label="新密码" prop="password">
26         <el-input type="password" v-model="chpwdForm.password"></el-
input>
27     </el-form-item>
28     <el-form-item label="确认新密码" prop="checkPass">
29         <el-input type="password" v-model="chpwdForm.checkPass"></el-
input>
30     </el-form-item>
31 </el-form>
32 <span slot="footer" class="dialog-footer">
33     <el-button @click="chpwdDialogVisible = false">取 消</el-button>
34     <el-button type="primary" @click="changePassword">确 定</el-
button>
35 </span>
36 </el-dialog>
37 </el-header>
38 <el-container>
39     <el-aside :width="isCollapsed ? '64px' : '200px'">
40         <el-menu
41             background-color="#123"
42             text-color="#fff"
43             active-text-color="#ffd04b"
44             :collapse="isCollapsed"
45             :collapse-transition="true"
46             :router="true"
47         >
48             <el-submenu v-for="item in menulist" :index="item.id + '"
:key="item.id">
49                 <template slot="title">
50                     <i class="el-icon-menu"></i><span slot="title">{{
item.itemName }}</span>
51                 </template>
52                 <el-menu-item v-for="subItem in item.children"
:index="subItem.path" :key="subItem.id">
53                     <i class="el-icon-menu"></i><span slot="title">{{
subItem.itemName }}</span>
54                 </el-menu-item>
55             </el-submenu>
56             <el-menu-item index="/welcome">
57                 <i class="el-icon-setting"></i>
58                 <span slot="title">导航四</span>
59             </el-menu-item>
60             <el-menu-item index="/t1">
61                 <i class="el-icon-setting"></i>
62                 <span slot="title">t1</span>
63             </el-menu-item>
64         </el-menu>
65     </el-aside>
66     <el-main>
67         <router-view></router-view>
68     </el-main>
69 </el-container>
70 </el-container>
71 </template>
72

```

```

73 <script>
74 export default {
75   created() {
76     this.getMenuList() // vue组件创建时查菜单数据
77     this.getUserInfo()
78   },
79   data() {
80     const validatePass = (rule, value, callback) => {
81       if (value !== this.chpwdForm.password) {
82         callback(new Error('两次输入密码不一致!'))
83       } else {
84         callback()
85       }
86     }
87     return {
88       menulist: [], // 菜单数据
89       isCollapsed: false,
90       user: {}, // 用户信息,
91       chpwdDialogVisible: false, // 修改密码对话框
92       chpwdForm: {
93         oldPassword: '',
94         password: '',
95         checkPass: ''
96       },
97       chpwdRules: {
98         oldPassword: [
99           { required: true, message: '请输入旧密码', trigger: 'blur' },
100           { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
101         ],
102         password: [
103           { required: true, message: '请输入新密码', trigger: 'blur' },
104           { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
105         ],
106         checkPass: [
107           { required: true, message: '请输入新密码', trigger: 'blur' },
108           { validator: validatePass, trigger: 'blur' }
109         ]
110       }
111     },
112     methods: {
113       async getMenuList() {
114         const { data: response } = await this.$http.get('users/menulist/')
115         if (response.code) {
116           return this.$message.error(response.message)
117         }
118         this.menulist = response.data
119       },
120       logout() {
121         window.localStorage.removeItem('token')
122         this.$router.push('/login')
123       },
124       toggleMenu() {
125         this.isCollapsed = !this.isCollapsed
126       },
127       handleCommand(command) {
128

```

```

129         if (command === 'logout') {
130             this.logout()
131         } else if (command === 'chpwd') {
132             this.chpwdDialogVisible = true
133         }
134     },
135     async getUserInfo() {
136         // 带着token发起请求, 后台查询用户数据
137         const { data: response } = await this.$http.get('users/whoami/')
138         if (response.code) {
139             return this.$message.error(response.message)
140         }
141         this.user = response.user
142     },
143     async changePassword() {
144         // 后台要验证旧密码
145         const { data: response } = await
146         this.$http.post(`users/${this.user.id}/setpwd/`, this.chpwdForm)
147         if (response.code) {
148             return this.$message.error(response.message)
149         }
150         this.chpwdDialogVisible = false
151         this.$message('密码修改成功')
152     },
153     resetForm(name) {
154         this.$refs[name].resetFields()
155     }
156 }
157 </script>

```

当前用户修改密码, 不要轻信user_id, 在后端, 一定要使用认证后的request.user。要求不能修改id为1的管理员密码。

utils/exceptions.py

```

1 class InvalidPassword(MagBaseException):
2     code = 101
3     message = '密码验证错误'

```

user/view.py完整代码

```

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
5 from django.contrib.auth import get_user_model
6 from .models import UserProfile
7 from .serializers import UserSerializer
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
13 class UserViewSet(ModelViewSet):

```

```

14 queryset = get_user_model().objects.all()
15 serializer_class = UserSerializer
16 # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
17 filter_backends = [filters.SearchFilter] # 指定filter
18 # filterset_fields = ['username']
19 search_fields = ['username']
20
21 # 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
22 def partial_update(self, request, *args, **kwargs):
23     print('~~~~@@@@@@@@@@@@')
24     print(request.user)
25     print(request.data)
26     print('~~~~@@@@@@@@@@@@')
27     request.data.pop('username', None) # 剔除不要更新的字段
28     request.data.pop('id', None)
29     request.data.pop('password', None)
30     return super().partial_update(request, *args, **kwargs)
31
32 def get_object(self):
33     if self.request.method.lower() != 'get':
34         pk = self.kwargs.get('pk')
35         if pk == 1 or pk == '1':
36             print('竟敢修改管理员!!!!')
37             raise Http404
38     return super().get_object()
39
40 @action(['GET'], detail=False, url_path='whoami')
41 def whoami(self, request): # detail=False, 非详情页
42     print(request.user)
43     return Response({
44         'user': {
45             'id': request.user.id,
46             'username': request.user.username
47         }
48     })
49
50 @action(detail=True, methods=['post'], url_path='setpwd')
51 def set_password(self, request, pk=None):
52     user = self.get_object()
53     # {'oldPassword': 'adminadmin', 'password': 'admin', 'checkPass':
'admin'}
54     if user.check_password(request.data['oldPassword']):
55         user.set_password(request.data['password'])
56         user.save()
57         return Response()
58     else:
59         raise InvalidPassword
60
61
62 class PermissionItem(dict):
63     def __init__(self, id, itemName, path=None):
64         super().__init__()
65         self.id = id
66         self.itemName = itemName
67         self.path = path
68         self.children = []
69
70     def __setattr__(self, key, value):

```

```

71         self[key] = value
72
73     def __getattr__(self, key):
74         return self[key]
75
76     def append(self, child):
77         self.children.append(child)
78
79 @api_view(['GET'])
80 @permission_classes([IsAuthenticated, IsAdminUser]) # 覆盖全局配置
81 def menulist_view(request: Request):
82     user = request.user
83     auth = request.auth
84     print(user, auth)
85     # print(dir(user)) # .is_superuser
86     menulist = {
87         'data': []
88     }
89     if user.is_superuser: # 用户管理必须管理员
90         p1 = PermissionItem(101, '用户管理')
91         p1.append(PermissionItem(102, '用户列表', '/users/'))
92         p1.append(PermissionItem(103, '角色列表', '/users/roles/'))
93         p1.append(PermissionItem(104, '权限列表', '/users/perms/'))
94         menulist['data'].append(p1)
95     return Response(menulist)

```

提交合并代码

前台代码提交

```

1 $ git add .
2 $ git commit -m "User finished"
3 $ git push -u origin user
4 $ git checkout master
5 $ git merge user
6 $ git push

```

后台代码提交

使用Pycharm图形界面操作，步骤同上。