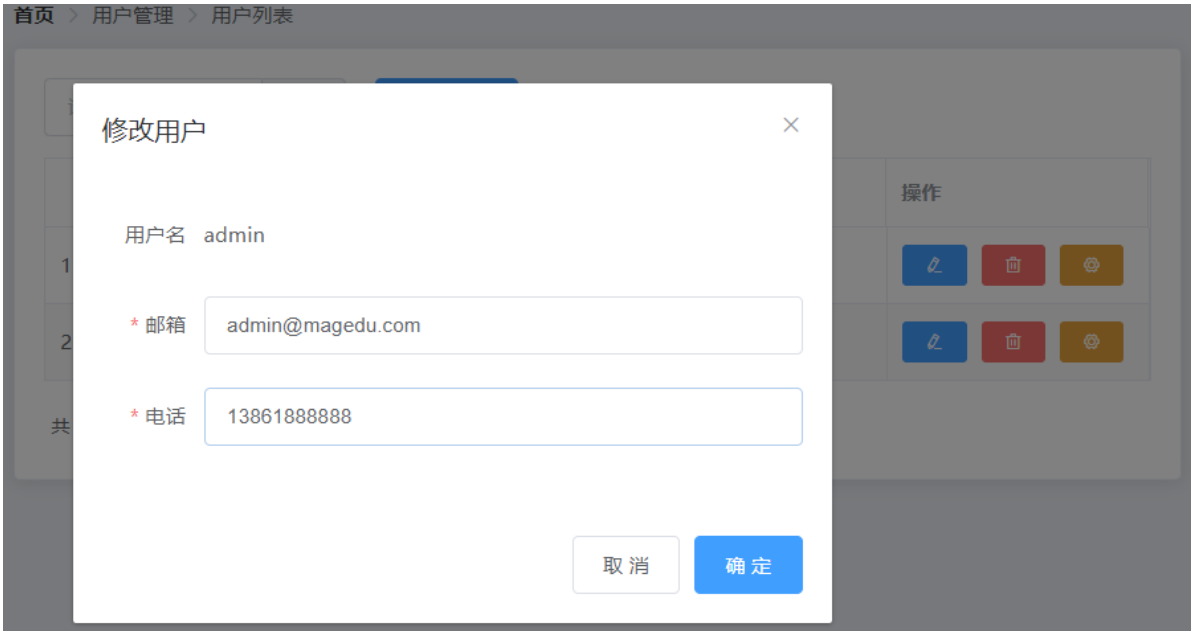
用户功能实现

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

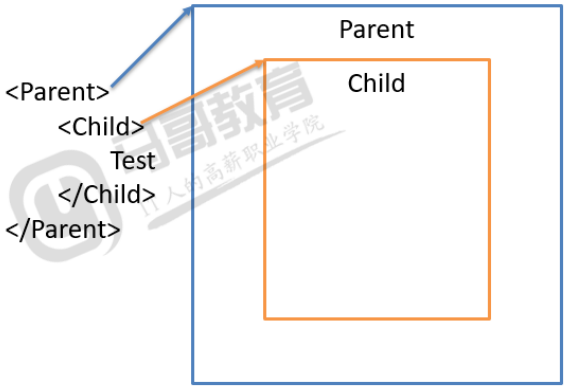


Vue插槽

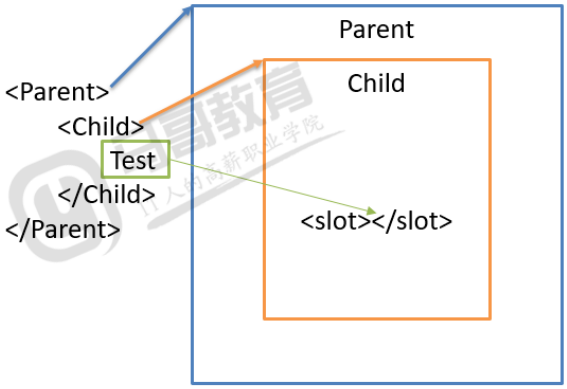
参考 https://cn.vuejs.org/v2/guide/components-slots.html

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。

```vue
<template>
  <div>
    父组件T1内容
    <hr />
    <T2>父组件中在子组件插入的内容c</T2>
  </div>
</template>

<script>
import T2 from './T2.vue'
export default {
  components: {
    T2
  }
}
</script>

<style>
</style>
```

```vue
<template>
  <div>子组件T2内容</div>
</template>

<script>
export default {}
</script>

<style>
</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

```vue
<template>
  <div>
    子组件T2内容
    <div style="border: 1px solid #000"><slot></slot></div>
  </div>
</template>

<script>
export default {}
</script>

<style>
</style>
```

父组件T1内容

子组件T2内容

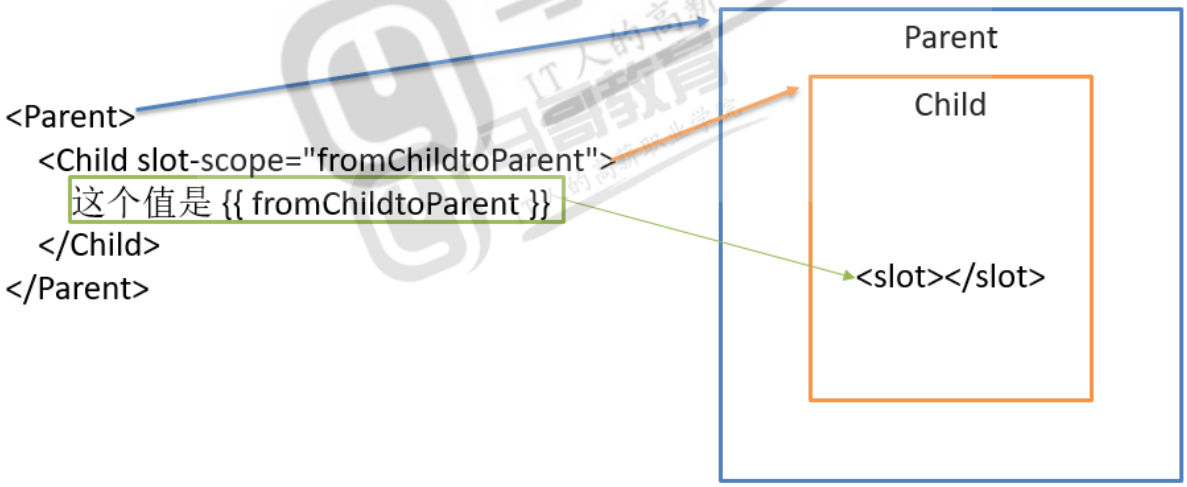
父组件中在子组件插入的内容c

> 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?

`作用域插座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

```
<template>
  <div>
    子组件T2内容
    <div style="border: 1px solid #000">
      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 />
      <slot :user="u"></slot>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      u: { name: 'tom', age: 20 }
    }
  }
}
</script>

<style>
</style>
```

修改父组件T1如下，如何使用子组件中的user?

```
<template>
  <div>
    父组件T1内容
    <hr />
    <T2>
      <template v-slot:default="xyz"> 从子组件中传来的数据是：{{ xyz.user.name }} </template>
    </T2>
  </div>
</template>

<script>
import T2 from './T2.vue'
export default {
  components: {
    T2
  }
}
</script>

<style>
</style>
```

可以继续简化为

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

顺便可以解构

```
<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

```
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
} from 'element-ui'

Vue.use(Switch)
```

采用下面的方式，不能把数据注入给switch组件

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

```
      <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="登录名"> </el-table-column>
        <el-table-column label="激活">
          <template #default="scope">
            <el-switch v-model="scope.row.is_active"></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">
          <el-button type="success" icon="el-icon-edit" size="mini"></el-
button>
          <el-button type="danger" icon="el-icon-delete" size="mini"></el-
button>
        </el-table-column>
```

```
      </el-table>
```

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

```
<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="input3">
              <el-button slot="append" icon="el-icon-search"></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-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">
          <el-button type="success" icon="el-icon-edit" size="mini"></el-button>
          <el-button type="danger" icon="el-icon-delete" size="mini"></el-button>
        </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>
    <!-- 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 {
      input3: '',
      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 }
    }
  },
  methods: {
    resetForm(name) {
      console.log('~~~~~~~~~~~~~~~~')
      this.$refs[name].resetFields()
    },
    addUser() {
      const name = 'addUser'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          const { data: response } = await this.$http.post('users/', this.addUserForm)
          if (response.code) {
            return this.$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.$http.get(`users/?page=${page}`)
      if (response.code) {
        return this.$message.error(response.message)
      }
      console.log(response)
      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/${row.id}/`, { is_active: row.is_active
})
    }
  }
}
</script>

<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。

```
axios.get('users/', {
    params: {
        page: 4
    }
})
```

所以，可以将getUserList(page)，修改如下

```
    async getUserList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('users/', {
        params: {
          page,
          username: this.search
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.userList = response.results
      this.pagination = response.pagination
    }
```

下面为了聚焦在改动的代码上，省略一些没改过的代码

```
<template>
  <div>
    <el-card class="box-card">
      <el-row :gutter="20">
```

```
        <el-col :span="12">
          <el-input placeholder="请输入内容" v-model="keywords">
            <el-button slot="append" icon="el-icon-search" @click="getUserList(1)"></el-button>
          </el-input>
        </el-col>
        <el-col :span="12">
          <el-button type="primary" @click="addUserDialogVisible = true">增加用户</el-button>
        </el-col>
      </el-row>
    </el-card>
  </div>
</template>

<script>
export default {
  created() {
    this.getUserList()
  },
  data() {
    return {
      keywords: '',
      // 用户列表表格
      userList: [],
      pagination: { page: 1, size: 20, total: 0 }
    }
  },
  methods: {
    async getUserList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get('users/', {
        params: {
          page,
          username: this.keywords
        }
      })
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.userList = response.results
      this.pagination = response.pagination
    }
  }
}
</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。

```
from rest_framework.viewsets import ModelViewSet
from django.contrib.auth import get_user_model
from .serializers import UserSerializer

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

    def get_queryset(self):
        queryset = super().get_queryset() # 调用父类的处理
        username = self.request.query_params.get('username', None)
        if username:
            queryset = queryset.filter(username__icontains=username)
        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

```
pip install django-filter
```

```
INSTALLED_APPS = [
    ...
    'django_filters',
    ...
]
```

```
REST_FRAMEWORK = {
    'DEFAULT_FILTER_BACKENDS':
['django_filters.rest_framework.DjangoFilterBackend']
}
```

```
from rest_framework.permissions import IsAuthenticated, IsAdminUser
from rest_framework.viewsets import ModelViewSet
from django.contrib.auth import get_user_model
from .models import UserProfile
from .serializers import UserSerializer
from django_filters.rest_framework import DjangoFilterBackend

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 = [DjangoFilterBackend] # 指定filter
    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

```python
from rest_framework import filters

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']
```

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

```vue
<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-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-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-button @click="handleEditUser(row)" type="success"
icon="el-icon-edit" size="mini"></el-button>
            <el-button type="danger" icon="el-icon-delete" size="mini">
</el-button>
          </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>
    <!-- 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>
  </div>
</template>

<script>
export default {
  data() {
    return {
      search: '',
      // 修改用户对话框
      editUserDialogVisible: false,
      editUserForm: {
        username: '',
        phone: '',
        first_name: '',
        last_name: '',
        email: ''
      },
      editUserRules: {}
    }
  },
  methods: {
    resetForm(name) {
      console.log('~~~~~~~~~~~~~~~~')
      this.$refs[name].resetFields()
    },
    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/${id}/`, this.editUserForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.editUserDialogVisible = false
          this.resetForm(name)
          this.getUserList(this.pagination.page) // 刷新用户列表
        }
      })
    }
  }
}
</script>

<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)`。

```python
from rest_framework.permissions import IsAuthenticated, IsAdminUser
from rest_framework.viewsets import ModelViewSet
from django.contrib.auth import get_user_model
from .models import UserProfile
from .serializers import UserSerializer
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework import filters

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):
        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)
```

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

```
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
} from 'element-ui'

Vue.use(Tooltip)

// 全局导入
Vue.prototype.$message = Message
Vue.prototype.$msgbox = MessageBox
```

```
      <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-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="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 type="danger" icon="el-icon-delete" size="mini"></el-button>
            </el-tooltip>
          </template>
        </el-table-column>
      </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。

```
<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="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>
    <!-- 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: {}
    }
  },
  methods: {
    resetForm(name) {
      console.log('~~~~~~~~~~~~~~~~')
```

```
      this.$refs[name].resetFields()
    },
    addUser() {
      const name = 'addUser'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          const { data: response } = await this.$http.post('users/', this.addUserForm)
          if (response.code) {
            return this.$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.$http.get('users/', {
        params: {
          page,
          search: this.search
        }
      })
      if (response.code) {
        return this.$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/${row.id}/`, { is_active: row.is_active })
    },
    handleEditUser(row) {
      this.editUserForm = row
      this.editUserDialogVisible = true
    },
    editUser() {
      const { id } = this.editUserForm
      const name = 'editUser'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          const { data: response } = await this.$http.patch(`users/${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/${id}/`)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.getUserList()
        })
        .catch(() => {})
    }
  }
}
</script>

<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?

```
from rest_framework.permissions import IsAuthenticated, IsAdminUser
from rest_framework.viewsets import ModelViewSet
from django.contrib.auth import get_user_model
from .models import UserProfile
from .serializers import UserSerializer
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework import filters
from django.http.response import Http404

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):
        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':
                raise Http404
        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

```
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
} from 'element-ui'

Vue.use(Dropdown)
Vue.use(DropdownMenu)
Vue.use(DropdownItem)
```

src/components/Home.vue

```
<template>
  <el-container>
    <el-header>
      <div class="logo">
        <img src="../assets/logo.png" alt="logo" />
        <div class="title">马哥教育猛犸运维系统管理平台</div>
        <div><i :class="isCollapsed ? 'el-icon-s-unfold' : 'el-icon-s-fold'" @click="toggleMenu"></i></div>
      </div>
      <div class="info">
        <el-button type="info" @click="logout">退出</el-button>
        <el-dropdown @command="handleCommand">
          <span class="el-dropdown-link">{{ user.username }}<i class="el-icon-arrow-down el-icon--right"></i> </span>
          <el-dropdown-menu slot="dropdown">
            <el-dropdown-item command="chpwd">修改密码</el-dropdown-item>
            <el-dropdown-item command="logout">退出</el-dropdown-item>
          </el-dropdown-menu>
        </el-dropdown>
      </div>
    </el-header>
    <el-container>
      <el-aside :width="isCollapsed ? '64px' : '200px'">
        <el-menu
          background-color="#123"
          text-color="#fff"
          active-text-color="#ffd04b"
```

```
          :collapse="isCollapsed"
          :collapse-transition="true"
          :router="true"
        >
          <el-submenu v-for="item in menulist" :index="item.id + ''"
:key="item.id">
            <template slot="title">
              <i class="el-icon-menu"></i><span slot="title">{{
item.itemName }}</span>
            </template>
            <el-menu-item v-for="subItem in item.children"
:index="subItem.path" :key="subItem.id">
              <i class="el-icon-menu"></i><span slot="title">{{
subItem.itemName }}</span>
            </el-menu-item>
          </el-submenu>
          <el-menu-item index="/welcome">
            <i class="el-icon-setting"></i>
            <span slot="title">导航四</span>
          </el-menu-item>
          <el-menu-item index="/t1">
            <i class="el-icon-setting"></i>
            <span slot="title">t1</span>
          </el-menu-item>
        </el-menu>
      </el-aside>
      <el-main>
        <router-view></router-view>
      </el-main>
    </el-container>
  </el-container>
</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

```
<script>
export default {
  created() {
    this.getMenulist() // vue组件创建时查菜单数据
    this.getUserInfo()
  },
  data() {
    return {
      menulist: [], // 菜单数据
      isCollapsed: false,
      user: {}, // 用户信息,
      chpwdDialogVisible: false // 修改密码对话框
    }
  },
  methods: {
    async getMenulist() {
      const { data: response } = await this.$http.get('users/menulist/')
      if (response.code) {
```

```
        return this.$message.error(response.message)
      }
      this.menulist = response.data
    },
    logout() {
      window.localStorage.removeItem('token')
      this.$router.push('/login')
    },
    toggleMenu() {
      this.isCollapsed = !this.isCollapsed
    },
    handleCommand(command) {
      if (command === 'logout') {
        this.logout()
      } else if (command === 'chpwd') {
        this.chpwdDialogVisible = true
      }
    },
    async getUserInfo() {
      // 带着token发起请求，后台查询用户数据
      const { data: response } = await this.$http.get('users/whoami/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.user = response.user
    },
    changePassword() {}
  }
}
</script>
```

user/views.py

```
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
from django.contrib.auth import get_user_model
from .models import UserProfile
from .serializers import UserSerializer
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework import filters
from django.http.response import Http404

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):
        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':
                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
            }
        })
```

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

```
<template>
  <el-container>
    <el-header>
      <div class="logo">
        <img src="../assets/logo.png" alt="logo" />
        <div class="title">马哥教育猛犸运维系统管理平台</div>
        <div><i :class="isCollapsed ? 'el-icon-s-unfold' : 'el-icon-s-fold'" @click="toggleMenu"></i></div>
      </div>
      <div class="info">
        <el-dropdown @command="handleCommand">
          <span class="el-dropdown-link">{{ user.username }}<i class="el-icon-arrow-down el-icon--right"></i> </span>
          <el-dropdown-menu slot="dropdown">
            <el-dropdown-item command="chpwd">修改密码</el-dropdown-item>
            <el-dropdown-item command="logout">退出</el-dropdown-item>
          </el-dropdown-menu>
        </el-dropdown>
      </div>
      <!-- 修改密码对话框 -->
      <el-dialog title="修改密码" :visible.sync="chpwdDialogVisible" @close="resetForm('chpwd')">
        <el-form :model="chpwdForm" :rules="chpwdRules" ref="chpwd" label-width="100px">
          <el-form-item label="用户名">{{ user.username }}</el-form-item>
          <el-form-item label="旧密码" prop="oldPassword">
```

```
            <el-input type="password" v-model="chpwdForm.oldPassword"></el-input>
          </el-form-item>
          <el-form-item label="新密码" prop="password">
            <el-input type="password" v-model="chpwdForm.password"></el-input>
          </el-form-item>
          <el-form-item label="确认新密码" prop="checkPass">
            <el-input type="password" v-model="chpwdForm.checkPass"></el-input>
          </el-form-item>
        </el-form>
        <span slot="footer" class="dialog-footer">
          <el-button @click="chpwdDialogVisible = false">取 消</el-button>
          <el-button type="primary" @click="changePassword">确 定</el-button>
        </span>
      </el-dialog>
    </el-header>
    <el-container>
      <el-aside :width="isCollapsed ? '64px' : '200px'">
        <el-menu
          background-color="#123"
          text-color="#fff"
          active-text-color="#ffd04b"
          :collapse="isCollapsed"
          :collapse-transition="true"
          :router="true"
        >
          <el-submenu v-for="item in menulist" :index="item.id + ''" :key="item.id">
            <template slot="title">
              <i class="el-icon-menu"></i><span slot="title">{{ item.itemName }}</span>
            </template>
            <el-menu-item v-for="subItem in item.children" :index="subItem.path" :key="subItem.id">
              <i class="el-icon-menu"></i><span slot="title">{{ subItem.itemName }}</span>
            </el-menu-item>
          </el-submenu>
          <el-menu-item index="/welcome">
            <i class="el-icon-setting"></i>
            <span slot="title">导航四</span>
          </el-menu-item>
          <el-menu-item index="/t1">
            <i class="el-icon-setting"></i>
            <span slot="title">t1</span>
          </el-menu-item>
        </el-menu>
      </el-aside>
      <el-main>
        <router-view></router-view>
      </el-main>
    </el-container>
  </el-container>
</template>

```

```
<script>
export default {
  created() {
    this.getMenulist() // vue组件创建时查菜单数据
    this.getUserInfo()
  },
  data() {
    const validatePass = (rule, value, callback) => {
      if (value !== this.chpwdForm.password) {
        callback(new Error('两次输入密码不一致!'))
      } else {
        callback()
      }
    }
    return {
      menulist: [], // 菜单数据
      isCollapsed: false,
      user: {}, // 用户信息,
      chpwdDialogVisible: false, // 修改密码对话框
      chpwdForm: {
        oldPassword: '',
        password: '',
        checkPass: ''
      },
      chpwdRules: {
        oldPassword: [
          { required: true, message: '请输入旧密码', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
        ],
        password: [
          { required: true, message: '请输入新密码', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
        ],
        checkPass: [
          { required: true, message: '请输入新密码', trigger: 'blur' },
          { validator: validatePass, trigger: 'blur' }
        ]
      }
    }
  },
  methods: {
    async getMenulist() {
      const { data: response } = await this.$http.get('users/menulist/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.menulist = response.data
    },
    logout() {
      window.localStorage.removeItem('token')
      this.$router.push('/login')
    },
    toggleMenu() {
      this.isCollapsed = !this.isCollapsed
    },
    handleCommand(command) {
```

```
      if (command === 'logout') {
        this.logout()
      } else if (command === 'chpwd') {
        this.chpwdDialogVisible = true
      }
    },
    async getUserInfo() {
      // 带着token发起请求，后台查询用户数据
      const { data: response } = await this.$http.get('users/whoami/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.user = response.user
    },
    async changePassword() {
      // 后台要验证旧密码
      const { data: response } = await
this.$http.post(`users/${this.user.id}/setpwd/`, this.chpwdForm)
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.chpwdDialogVisible = false
      this.$message('密码修改成功')
    },
    resetForm(name) {
      this.$refs[name].resetFields()
    }
  }
}
</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

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

user/view.py完整代码

```
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
from django.contrib.auth import get_user_model
from .models import UserProfile
from .serializers import UserSerializer
from django_filters.rest_framework import DjangoFilterBackend
from rest_framework import filters
from django.http.response import Http404
from utils.exceptions import InvalidPassword

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


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)

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

提交合并代码

前台代码提交

```
$ git add .
$ git commit -m "User finished"
$ git push -u origin user
$ git checkout master
$ git merge user
$ git push
```

后台代码提交

使用Pycharm图形界面操作，步骤同上。