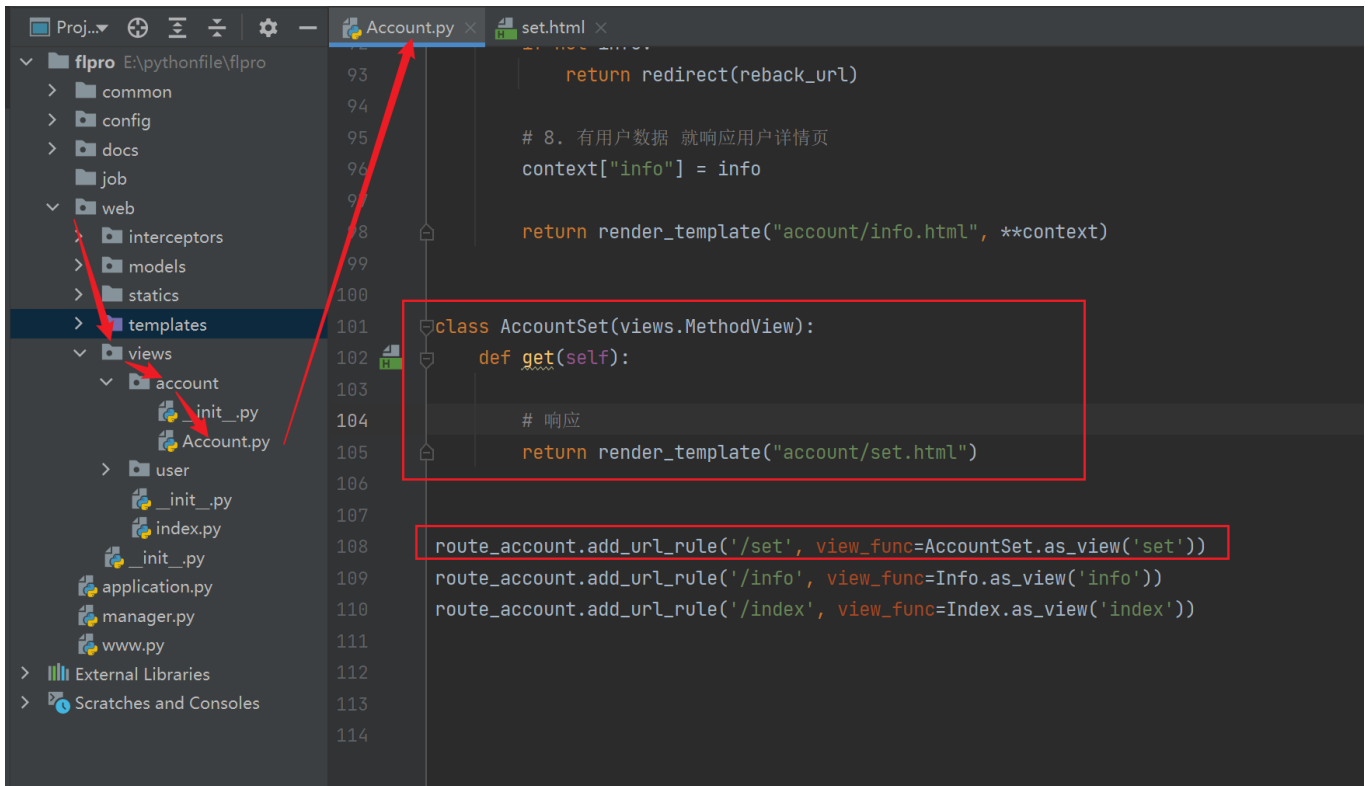


1.渲染编辑页



```
# -*- encoding: utf-8 -*-
"""
File      : Account.py
teaching  :
    用户列表
"""
from flask import Blueprint, render_template, views, request, jsonify, make_response, url_for,
redirect, g
from common.lib.UrlManager import UrlManager
from web.models.User import User # 模型类导出
from common.lib.UserService import UserService
from application import app, db
import json
from common.lib.Helper import iPagination

# 构建蓝图对象
route_account = Blueprint('account_page', __name__)

class Index(views.MethodView):

    def get(self):
        # 1.获取查询用户数据对象
        query = User.query

        # 2.获取请求中的数据
        req = request.values

        # 3.获取请求携带的当前页码
```

```

page = int(req["p"]) if ("p" in req and req["p"]) else 1 # 当前第几页, 默认1

# 4.获取请求路由
# full_path 携带查询字符串的路由数据 ---> /account/index?
full_url = request.full_path
# print(full_url)
url = full_url.replace(f"&p={page}", "") # 不要查询字符串数据 --> &p={page} 相当于剩下/account/index?

# 4.用于分页的参数
page_params = {
    'total': query.count(), # 数据总数
    'page_size': app.config["PAGE_SIZE"], # 每页数据量
    'display': app.config["PAGE_DISPLAY"], # 显示页码栏数量
    'page': page, # 当前页码
    'url': url # /account/index?
}

# from common.lib.Helper import iPagination
# 5.调用分页器
pages = iPagination(page_params)

# 6. 计算展示的 起始数据 比如展示第2 (page) 页数据 每页展示20 (PAGE_SIZE) 条数据, 那么第二页就是起始位置 20
offset = (page - 1) * app.config["PAGE_SIZE"]

# 7. 计算展示的 结束数据 比如展示第2页数据 每页展示20条数据, 那么第二页就是结束位置 40
limit = page * app.config["PAGE_SIZE"]

# 8.查询所有数据
user_all = query.order_by(User.uid.asc()).all() # 获取所有的用户数据, 并根据uid进行升序排列
asc 降序就是desc

# 9. 构建响应数据
context = {
    "list": user_all[offset:limit], # 展示数据信息列表
    "pages": pages,
}

# 10.渲染响应
return render_template('account/index.html', **context)

class Info(views.MethodView):
    def get(self):

        # 1.建立渲染数据字典
        context = {}

        # 2.获取请求携带的查询字符串数据
        req = request.args

        # 3.获取查询字符串 id键数据
        uid = int(req.get("id", 0)) # 没有就设置默认值0, 代表没有

        # 4.构建请求路由 -- 列表页
        reback_url = UrlManager.buildUrl("/account/index")

        # 5. uid 小于1 代表没有该用户数据

```

```

if uid < 1:
    return redirect(reback_url) # 回到列表页

# 6.根据uid查询对应用户 -- 获取用户数据
info = User.query.filter_by(uid=uid).first()

# 7. 如果没有用户数据 回到列表页
if not info:
    return redirect(reback_url)

# 8. 有用户数据 就响应用户详情页
context["info"] = info

return render_template("account/info.html", **context)

class AccountSet.views.MethodView:
    def get(self):
        return render_template("account/set.html")

route_account.add_url_rule('/set', view_func=AccountSet.as_view('set'))
route_account.add_url_rule('/info', view_func=Info.as_view('info'))
route_account.add_url_rule('/index', view_func=Index.as_view('index'))

```

启动访问 <http://127.0.0.1:8888/account/index>

欢迎使用编程浪子订餐管理后台

账户列表

请选择状态 请输入姓名或者手机号码 搜索

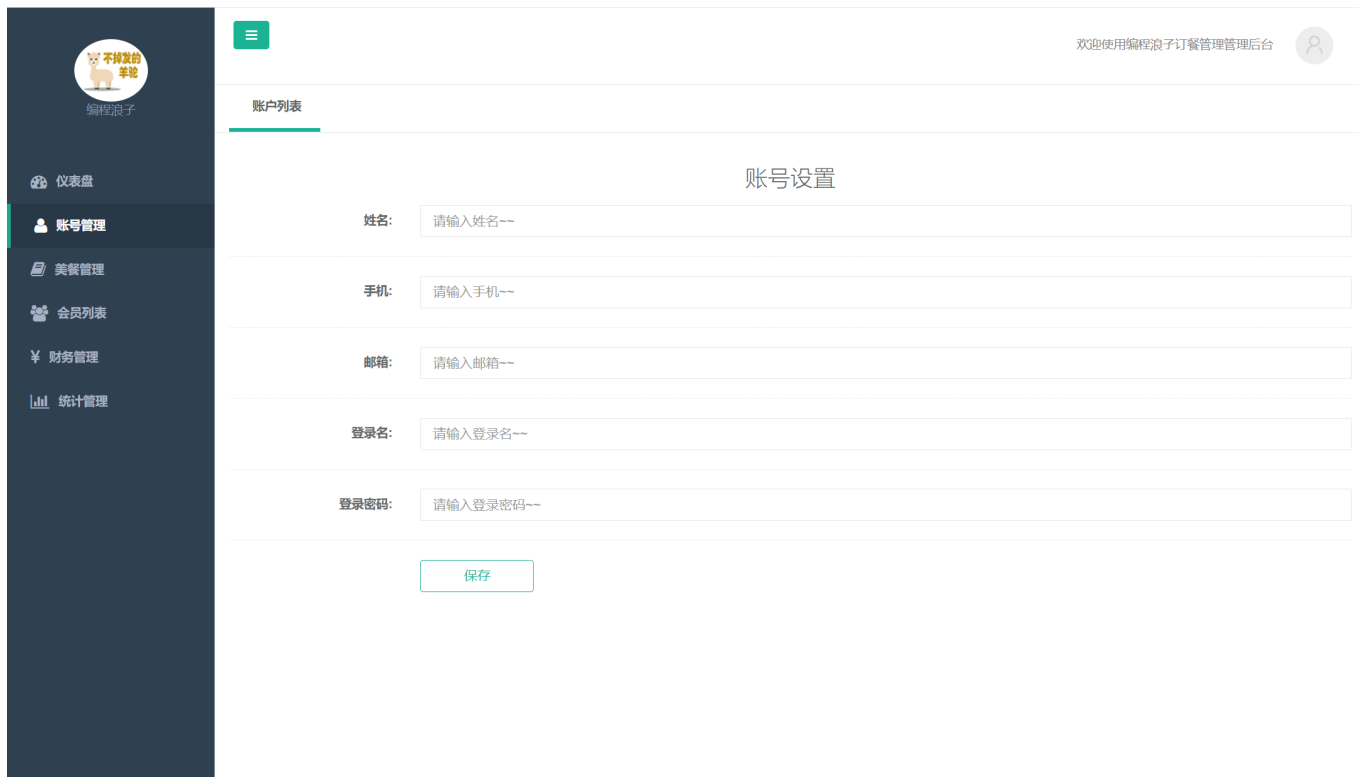
序号	姓名	手机	邮箱	操作
1	风勋	11012345679	111222@163.com	编辑 删除

共1条记录 | 每页20条

1

编辑页--添加账号

编辑页, 编辑账户信息



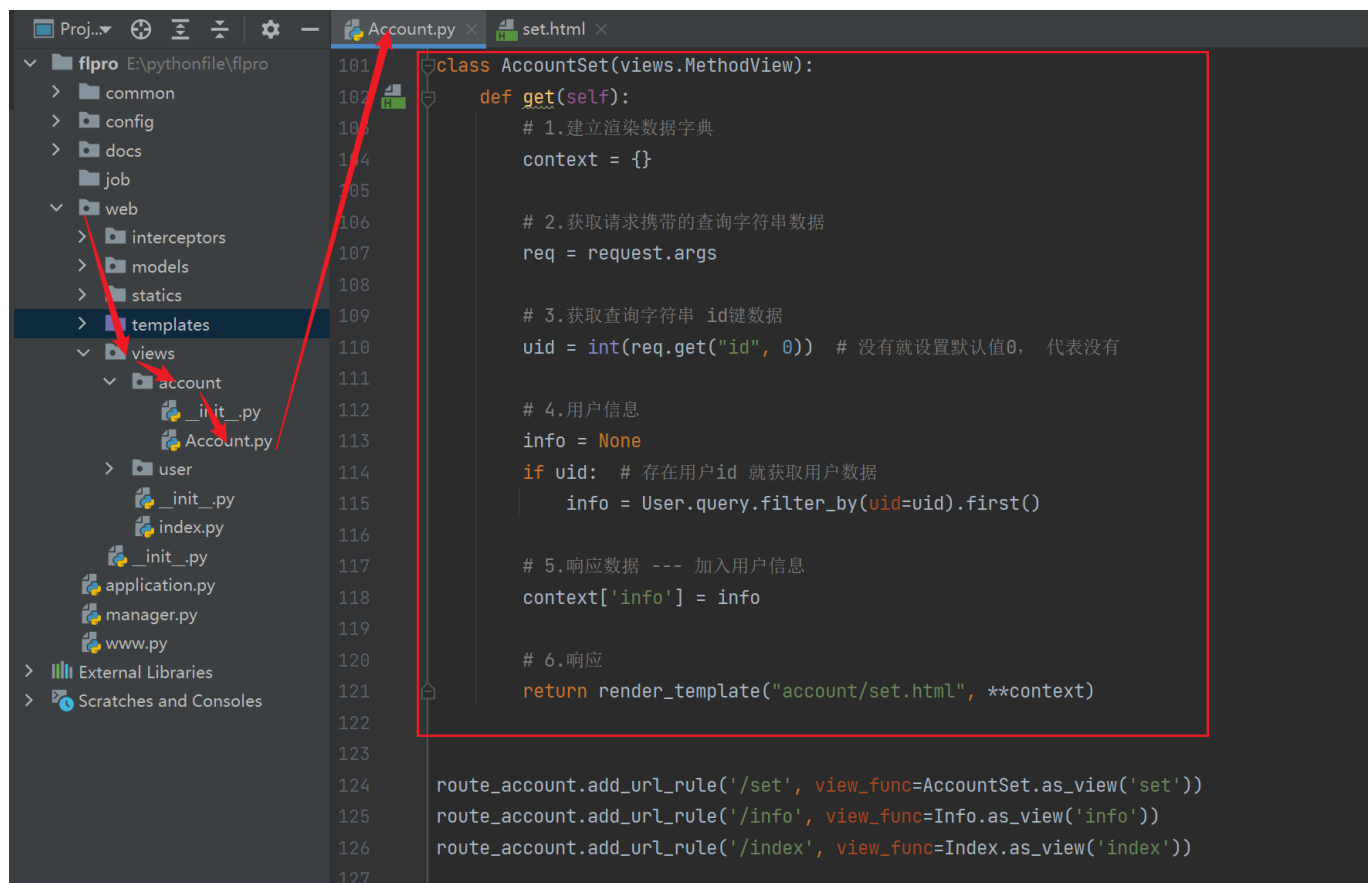
2.编辑页信息

编辑页作用：

- 1.账号编辑页 --- 需要渲染用户原来的信息
- 2.账号创建页 --- 不需要渲染用户原来信息

首先大家需要搞清楚， 渲染账户信息，需要用户uid信息，没有就无法获取用户信息

那么在渲染时，我们就只需通过判断请求信息是否携带uid信息 来分编是新增页还是编辑页



```
class AccountSet(views.MethodView):
    def get(self):
        # 1.建立渲染数据字典
        context = {}

        # 2.获取请求携带的查询字符串数据
        req = request.args

        # 3.获取查询字符串 id键数据
        uid = int(req.get("id", 0)) # 没有就设置默认值0, 代表没有

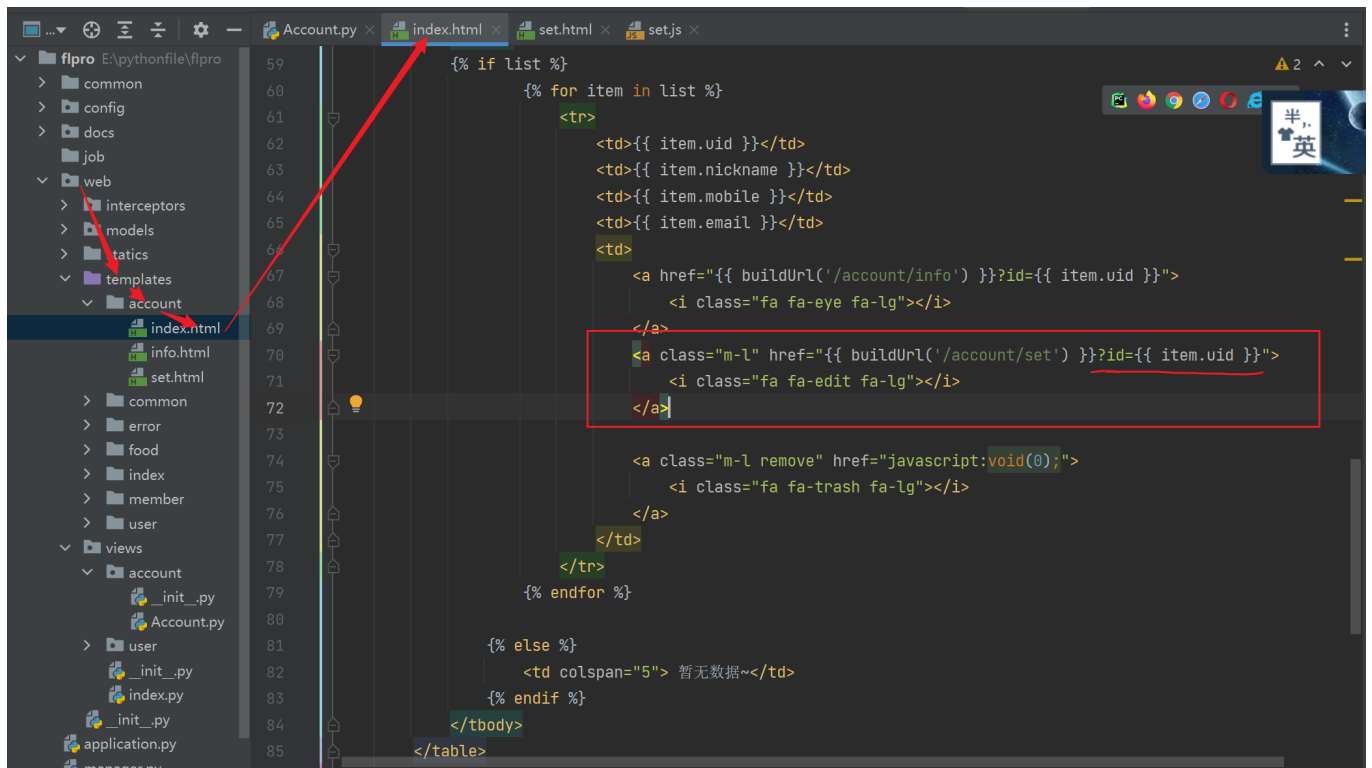
        # 4.用户信息
        info = None
        if uid: # 存在用户id 就获取用户数据 --- 即编辑页; 反之为新增页
            info = User.query.filter_by(uid=uid).first()

        # 5.响应数据 --- 加入用户信息
        context['info'] = info

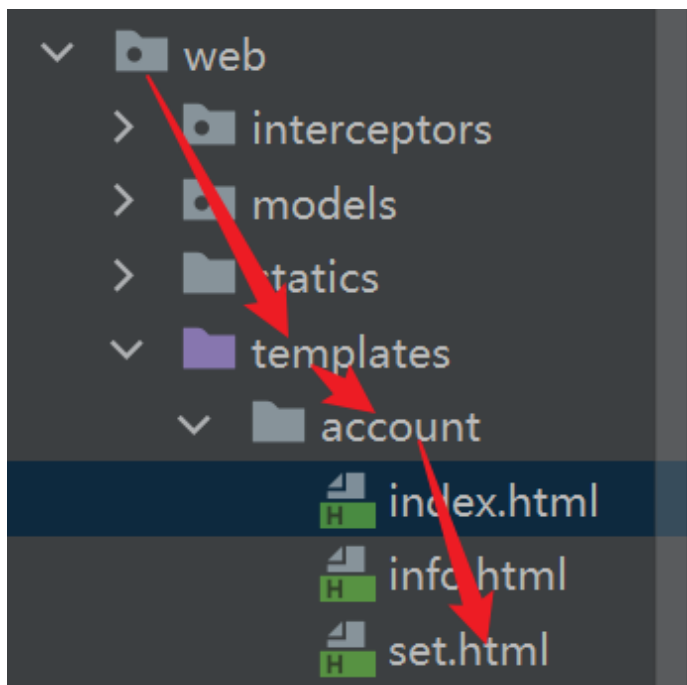
        # 6.响应
        return render_template("account/set.html", **context)
```

此时前端模板也需要调整

--index.html--



--set.html--




```
Account.py x index.html x set.html x set.js x
41 <div class="col-lg-10">
42 <input type="text" name="login_name" class="form-control" autocomplete="off" placeholder="请输入登录名~~" value="" />
43 </div>
44 </div>
45 <div class="hr-line-dashed"></div>
46 <div class="form-group">
47 <label class="col-lg-2 control-label">登录密码:</label>
48 <div class="col-lg-10">
49 <input type="password" name="login_pwd" class="form-control" autocomplete="new-password" placeholder="请输入登录密码~~" value="" />
50 </div>
51 </div>
52 <div class="hr-line-dashed"></div>
53 <div class="form-group">
54 <div class="col-lq-4 col-lq-offset-2">
55 <input type="hidden" name="id" value="{{ info.uid }}" /> {# 隐藏标识, 如果有id则表示市编辑账号, 否则新增账号 #}
56 <button class="btn btn-w-m btn-outline btn-primary save">保存</button>
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 {% endblock %}
63 {% block js %}
64 <script src="{{ buildStaticUrl('/js/account/set.js') }}" /></script>
65 {% endblock %}
```

```
{% extends "common/layout_main.html" %}
{% block content %}
<div class="row border-bottom">
  <div class="col-lg-12">
    <div class="tab_title">
      <ul class="nav nav-pills">
        <li class="current">
          <a href="{{ buildUrl('/account/index') }}" />账户列表</a>
        </li>
      </ul>
    </div>
  </div>
</div>
<div class="row m-t wrap_account_set">
  <div class="col-lg-12">
    <h2 class="text-center">账号设置</h2>
    <div class="form-horizontal m-t m-b">
      <div class="form-group">
        <label class="col-lg-2 control-label">姓名:</label>
        <div class="col-lg-10">
          <input type="text" name="nickname" class="form-control" placeholder="请输入姓名~~" value="{{ info.nickname }}" />
        </div>
      </div>
      <div class="hr-line-dashed"></div>
      <div class="form-group">
        <label class="col-lg-2 control-label">手机:</label>
        <div class="col-lg-10">
          <input type="text" name="mobile" class="form-control" placeholder="请输入手机~~" value="{{ info.mobile }}" />
        </div>
      </div>
      <div class="hr-line-dashed"></div>
      <div class="form-group">
        <label class="col-lg-2 control-label">邮箱:</label>
      </div>
    </div>
  </div>
</div>
```

```

    <div class="col-lg-10">
      <input type="text" name="email" class="form-control" placeholder="请输入邮箱~~" value="{{
info.email }}">
    </div>
  </div>
  <div class="hr-line-dashed"></div>
  <div class="form-group">
    <label class="col-lg-2 control-label">登录名:</label>
    <div class="col-lg-10">
      <input type="text" name="login_name" class="form-control" autocomplete="off" placeholder="请输
入登录名~~" value="{{ info.login_name }}">
    </div>
  </div>
  <div class="hr-line-dashed"></div>
  <div class="form-group">
    <label class="col-lg-2 control-label">登录密码:</label>
    <div class="col-lg-10">
      <input type="password" name="login_pwd" class="form-control" autocomplete="new-
password" placeholder="请输入登录密码
~~" {% if info %} value="*****" {% else %} value="" {% endif %}>
    </div>
  </div>
  <div class="hr-line-dashed"></div>
  <div class="form-group">
    <div class="col-lg-4 col-lg-offset-2">
      <input type="hidden" name="id" value="{{ info.uid }}"> {# 隐藏标识, 如果有id则表
示市编辑账号, 否则新增账号 #}
      <button class="btn btn-w-m btn-outline btn-primary save">保存</button>
    </div>
  </div>
</div>
{% endblock %}
{%block js %}
  <script src="{{ buildStaticUrl('/js/account/set.js') }}"></script>
{% endblock %}

```

启动 访问 <http://127.0.0.1:8888/account/index>

仪表盘

账号管理

美食管理

会员列表

财务管理

统计管理

菜单

欢迎使用编程浪子订餐管理后台

用户头像

账号列表

请选择状态

请输入姓名或者手机号码

搜索

+账号

序号	姓名	手机	邮箱	操作
1	风勋	11012345679	111222@163.com	<div><div>眼睛图标</div><div>编辑图标</div><div>删除图标</div></div>

共1条记录 | 每页20条

1

仪表盘

账号管理

美食管理

会员列表

财务管理

统计管理

菜单

欢迎使用编程浪子订餐管理后台

用户头像

账号列表

账号设置

姓名: 风勋

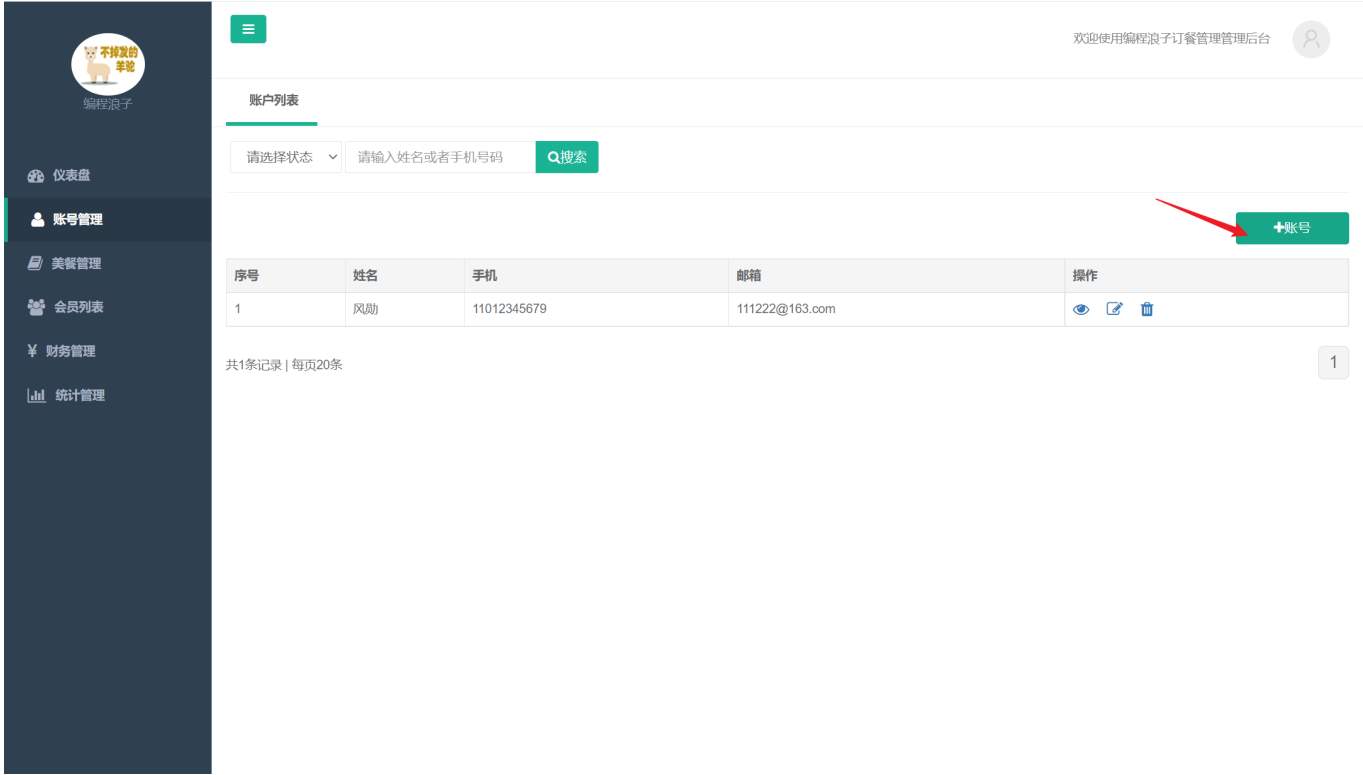
手机: 11012345679

邮箱: 111222@163.com

登录名: fengxun

登录密码:

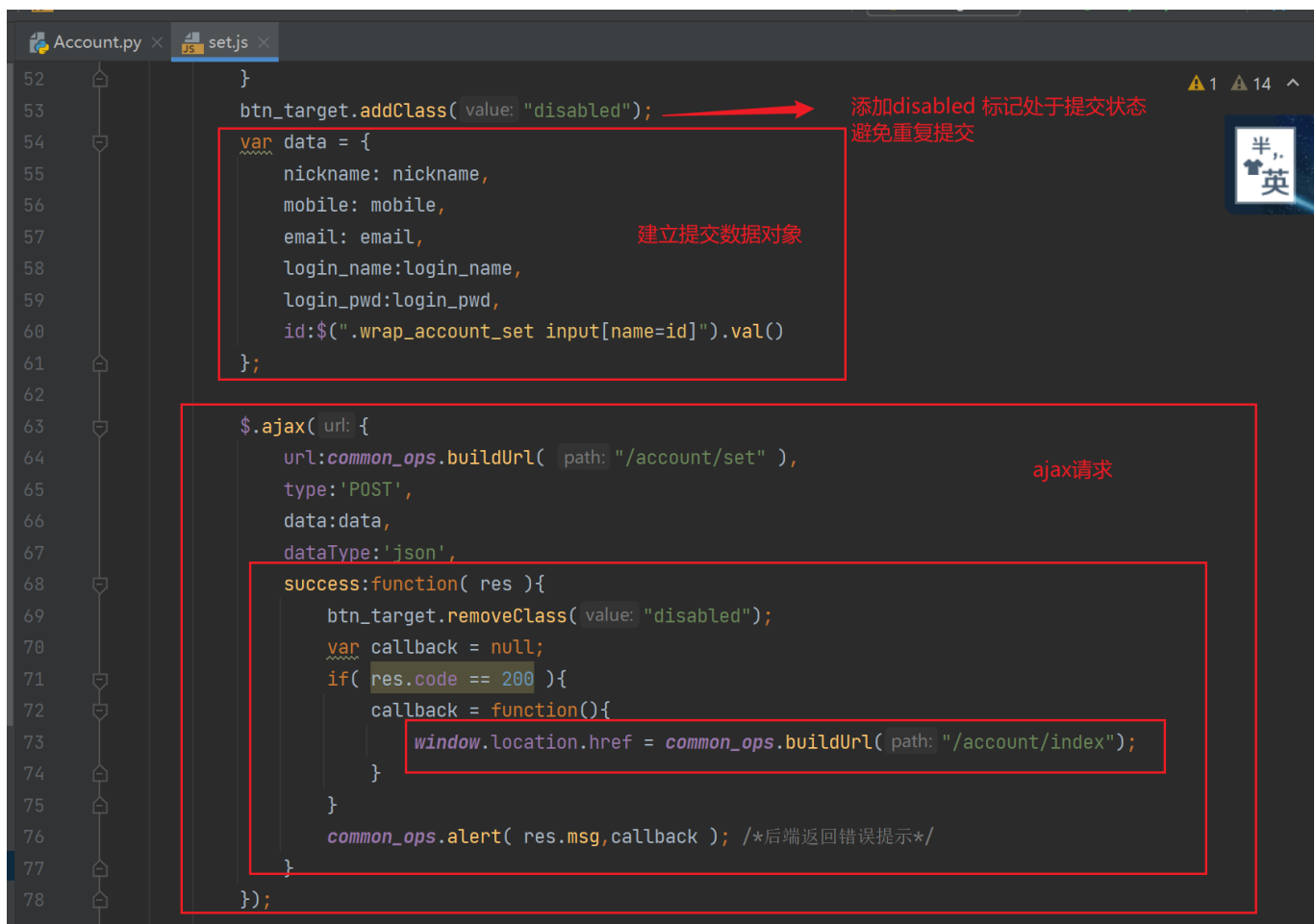
保存



3.编辑与新建实现 -- js分析

```
4 this.eventBind();
5 },
6 eventBind:function(){
7     $(".wrap_account_set .save").click(function(){ →点击保存 -- 触发点击事件
8         var btn_target = $(this);
9         if( btn_target.hasClass( selector: "disabled" ) ){
10             common_ops.alert( msg: "正在处理!!请不要重复提交~~");
11             return;
12         }
13         var nickname_target = $(".wrap_account_set input[name=nickname"]);
14         var nickname = nickname_target.val();
15         var mobile_target = $(".wrap_account_set input[name=mobile"]);
16         var mobile = mobile_target.val();
17         var email_target = $(".wrap_account_set input[name=email"]);
18         var email = email_target.val();
19         var login_name_target = $(".wrap_account_set input[name=login_name"]);
20         var login_name = login_name_target.val();
21         var login_pwd_target = $(".wrap_account_set input[name=login_pwd"]);
22         var login_pwd = login_pwd_target.val();
23         if( nickname.length < 1 ){
24             common_ops.tip( msg: "请输入符合规范的姓名~~",nickname_target );
25         }
26     }
27 }
28
29
30
```

```
8
9 if( nickname.length < 1 ){
10     common_ops.tip( msg: "请输入符合规范的姓名~~",nickname_target );
11     return false;
12 }
13
14 if( mobile.length < 1 ){
15     common_ops.tip( msg: "请输入符合规范的手机号码~~",mobile_target );
16     return false;
17 }
18
19 if( email.length < 1 ){
20     common_ops.tip( msg: "请输入符合规范的邮箱~~",email_target );
21     return false;
22 }
23
24 if( login_name.length < 1 ){
25     common_ops.tip( msg: "请输入符合规范的登录用户名~~",login_name_target );
26     return false;
27 }
28
29 if( login_pwd.length < 6 ){
30     common_ops.tip( msg: "请输入符合规范的登录密码~~",login_pwd_target );
31     return false;
32 }
33
34 btn_target.addClass( value: "disabled");
35 var data = {
36     account_set_ops: eventBind(),
37     callback: $(".wrap_account_set .save").click()
38 }
39
```



5.编辑与新建实现

首先因为我们这里新增账户需要生成加密盐，因此我们需要建立一个专门生成加密盐的方法

common/lib/UrlManager.py 代码如下 (genSalt)

```
# -*-coding:utf-8-*-
import hashlib, base64
import string
import random

class UserService():

    @staticmethod
    def genPwd(pwd, salt):
        """
        密码加密
        :param pwd: 密码
        :param salt: 加密盐
        :return: 加密
        """

        # base64加密
        pwd_base64 = base64.encodebytes(pwd.encode("utf-8")) # 将 密码 bs64加密
        data_str = f"{pwd_base64}-{salt}" # 拼接 加密密码数据 与盐拼接
```

```

# md5加密
m = hashlib.md5() # 实例md5加密对象
m.update(data_str.encode("utf-8")) # 加密
return m.hexdigest() # 获取加密数据

@staticmethod
def genAuthCode(user_info):
    """
    根据用户实例对象生成密钥
    :param user_info: 用户对象
    :return:
    """

    # 用户对象的 id 用户名 用户密码 用户盐 拼接构建加密字符串
    str = f"{user_info.uid}-{user_info.login_name}-{user_info.login_pwd}-{user_info.login_salt}"
    # md5加密
    m = hashlib.md5()
    m.update(str.encode("utf-8"))
    return m.hexdigest() # 获取加密数据

@staticmethod
def genSalt(length=16):
    """生成 授权码"""
    # 1.获取密码字符集
    # import string
    # string.ascii_letters 大小写字母
    # string.digits 数字
    selectString = string.ascii_letters + string.digits # 所有的数字和字母

    # 随机生成16个数字或者字母的组合
    key_list = [random.choice(selectString) for i in range(length)]
    return "".join(key_list)

```

其次 更新时间以及创建时间 均需获取当前时间，我们也可以封装一个独立方法专门获取当前时间

common/lib/Helper.py 代码如下

```

# 获取当前时间
def getCurrentDate(format="%Y-%m-%d %H:%M:%S"):
    # return datetime.datetime.now().strftime(format) #返回字符串格式
    return datetime.datetime.now() # 返回时间格式

```

实现编辑及新增逻辑

```
class AccountSet.views.MethodView):
    def get(self):
        # 1. 建立渲染数据字典
        context = {}

        # 2. 获取请求携带的查询字符串数据
        req = request.args

        # 3. 获取查询字符串 id键数据
        uid = int(req.get("id", 0)) # 没有就设置默认值0, 代表没有

        # 4. 用户信息
        info = None
        if uid: # 存在用户id 就获取用户数据
            info = User.query.filter_by(uid=uid).first()

        # 5. 响应数据 --- 加入用户信息
        context['info'] = info

        # 6. 响应
        return render_template("account/set.html", **context)

    def post(self):
        # 1. 响应数据对象构建
        resp = {"code": 200, "msg": "操作成功", "data": {}}

        # 2. 获取请求 提交的数据对象
        req = request.values

        # 3. 获取数据 注意 忘记if else 的三目运算的同学回顾基础
        nickname = req["nickname"] if "nickname" in req else ""
        mobile = req["mobile"] if "mobile" in req else ""
        email = req["email"] if "email" in req else ""
        login_name = req["login_name"] if "login_name" in req else ""
        login_pwd = req["login_pwd"] if "login_pwd" in req else ""
        id = req["id"] if "id" in req else 0

        # 4. 校验数据是否都获取到数据 及 数据是否符合要求
        if (not nickname) or len(nickname) < 1:
            resp["code"] = -1
            resp["msg"] = "请输入符合规范的姓名"
        if (not mobile) or len(mobile) < 1:
            resp["code"] = -1
            resp["msg"] = "请输入符合规范的手机号码"
```

```
class AccountSet.views.MethodView):
    def get(self):
        # 1. 建立渲染数据字典
        context = {}

        # 2. 获取请求携带的查询字符串数据
        req = request.args

        # 3. 获取查询字符串 id键数据
        uid = int(req.get("id", 0)) # 没有就设置默认值0, 代表没有

        # 4. 用户信息
        info = None
        if uid: # 存在用户id 就获取用户数据
            info = User.query.filter_by(uid=uid).first()

        # 5. 响应数据 --- 加入用户信息
        context['info'] = info

        # 6. 响应
        return render_template("account/set.html", **context)

    def post(self):
        # 1. 响应数据对象构建
        resp = {"code": 200, "msg": "操作成功", "data": {}}

        # 2. 获取请求 提交的数据对象
        req = request.values

        # 3. 获取数据 注意 忘记if else 的三目运算的同学回顾基础
        nickname = req["nickname"] if "nickname" in req else ""
```

```

mobile = req["mobile"] if "mobile" in req else ""
email = req["email"] if "email" in req else ""
login_name = req["login_name"] if "login_name" in req else ""
login_pwd = req["login_pwd"] if "login_pwd" in req else ""
id = req["id"] if "id" in req else 0

# 4. 校验数据是否都获取到数据 及 数据是否符合要求
if (not nickname) or len(nickname) < 1:
    resp["code"] = -1
    resp["msg"] = "请输入符合规范的姓名"
if (not mobile) or len(mobile) < 1:
    resp["code"] = -1
    resp["msg"] = "请输入符合规范的手机号码"
    return jsonify(resp)
if (not email) or len(email) < 1:
    resp["code"] = -1
    resp["msg"] = "请输入符合规范的邮箱"
    return jsonify(resp)
if (not login_name) or len(login_name) < 1:
    resp["code"] = -1
    resp["msg"] = "请输入符合规范的登录名 "
    return jsonify(resp)
if (not login_pwd) or len(login_pwd) < 6:
    resp["code"] = -1
    resp["msg"] = "请输入符合规范的密码"
    return jsonify(resp)

# 5.验证登录名是否被注册 在添加账号时, User.uid != id才会成立
# User.login_name == login_name 查看是否已注册,
# User.uid != id 这是因为如果是编辑账户信息User.uid == id 的, 所以这是为了区分编辑 --- 添
加账号 User.uid != id, 因为新增用户压根不存在于数据库
has_in = User.query.filter(User.login_name == login_name, User.uid != id).first()
if has_in: # 如果查出用户信息 -- 说明新增账户的登录名已经存在了
    resp["code"] = -1
    resp["msg"] = "该登录名已经存在, 请换一个试一试 "
    return jsonify(resp)

# 6.编辑or添加账号?
user_info = User.query.filter_by(uid=id).first() # 根据id信息获取用户数据
if user_info: # 能拿到就是编辑请求
    # 如果admin账号, 不允许修改
    if user_info and user_info.uid == 1:
        resp["code"] = -1
        resp["msg"] = "该用户是admin账号, 不允许修改编辑 "
        return jsonify(resp)
    model_user = user_info

else: # 拿不到就是新增账号请求
    model_user = User() # 新建一个User实例
    # UserService.genSalt(16) -- 生成加密盐
    model_user.login_salt = UserService.genSalt(16)

    # from common.lib.Helper import getCurrentDate
    # getCurrentDate()获取时间对象
    model_user.created_time = getCurrentDate() # 创建时间

# 7.更新用户对象数据
model_user.nickname = nickname
model_user.mobile = mobile

```

```

model_user.email = email
model_user.login_name = login_name

# 8.是否重设密码 --- 密码修改有独立的页面进行, 此处不能进行密码修改
if login_pwd != '*****': # login_pwd != '*****' 表示修改了密码 重新生成密码保存
    # 通过密码及加密盐 生成 加密密码
    model_user.login_pwd = UserService.genPwd(login_pwd, model_user.login_salt)


# 9.设置更新时间
model_user.updated_time = getCurrentDate()


# 10.提交保存数据库
db.session.add(model_user)
db.session.commit()

# 响应成功
return jsonify(resp)


```

启动访问 <http://127.0.0.1:8888/account/index>





欢迎使用编程浪子订餐管理后台
 




账户列表

请选择状态
 

请输入姓名或者手机号码

 搜索

 账号

序号	姓名	手机	邮箱	操作
1	风励	11012345679	111222@163.com	  

共1条记录 | 每页20条

1

账号设置

姓名: 帅风

手机: 18874203970

邮箱: 1234@qq.com

登录名: fengfeng

登录密码: ***** 123456

保存



- 仪表盘
- 账号管理
- 美食管理
- 会员列表
- 财务管理
- 统计管理



账户列表

欢迎使用编程浪子订餐管理后台



账号设置

姓名: 帅风

手机: 18874203970

邮箱: 1234@qq.com

登录名: fengfeng

登录密码: *****

保存

信息



操作成功

确定

欢迎使用编程浪子订餐管理后台

账户列表

请选择状态

请输入姓名或者手机号码

搜索

+账号

序号	姓名	手机	邮箱	操作
1	风勋	11012345679	111222@163.com	<div><div></div><div></div><div></div></div>
2	帅风	18874203970	1234@qq.com	<div><div></div><div></div><div></div></div>

共2条记录 | 每页20条

1

6.删除与恢复账号 -- js分析

账户列表

请选择状态

请输入姓名或者手机号码

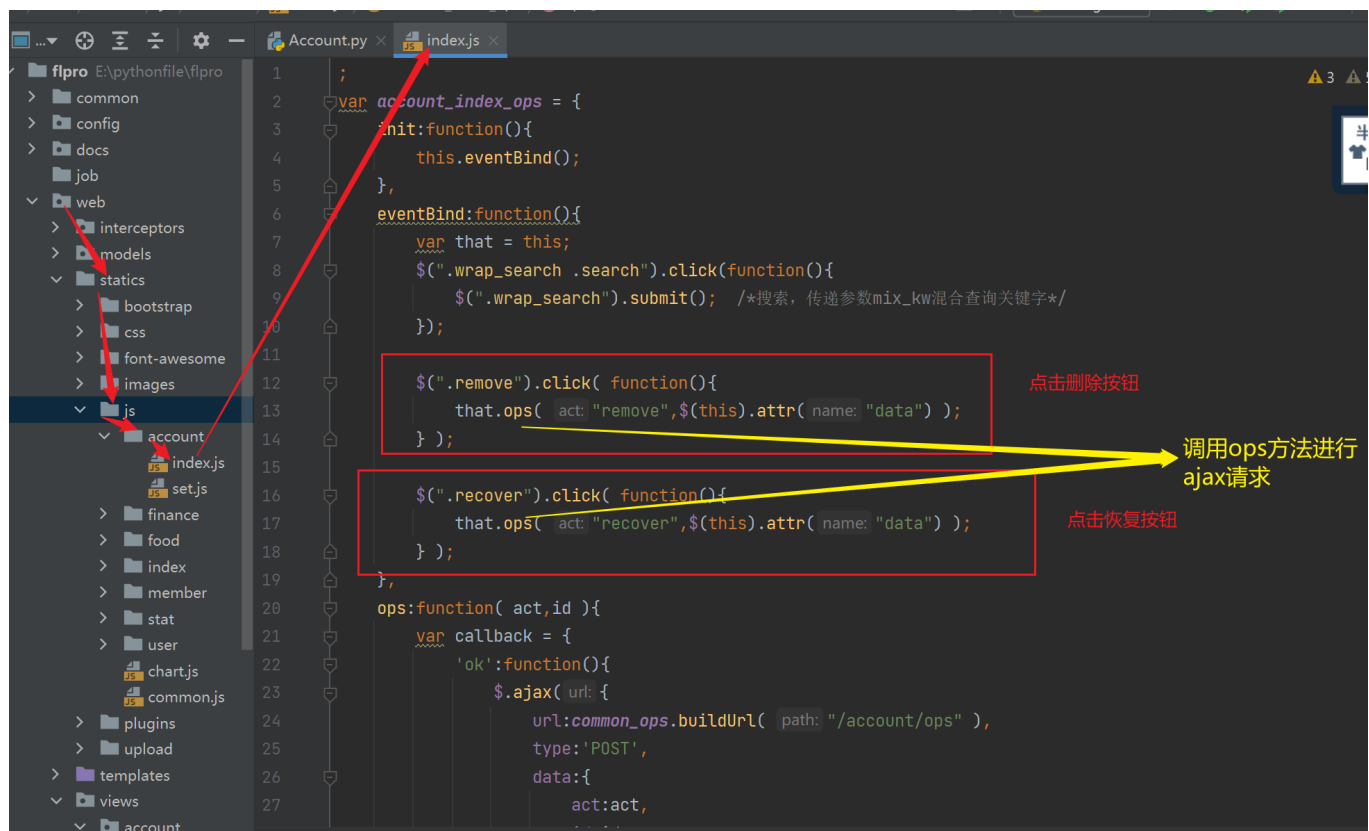
搜索

+账号

序号	姓名	手机	邮箱	操作
1	风勋	11012345679	111222@163.com	<div><div></div><div></div><div></div></div>
2	帅风	18874203970	1234@qq.com	<div><div></div><div></div><div></div></div>

共2条记录 | 每页20条

点击后



7.删除与恢复账号


```

# 4. 检验act参数 是不是 记录的标准操作字段  remove--删除  recover--恢复
if act not in ["remove", "recover"]:
    resp["code"] = -1
    resp["msg"] = "操作有误"
    return jsonify(resp)

# 5.检验操作 数据是否获取到
if not id:
    resp["code"] = -1
    resp["msg"] = "请选择要操作的账号"
    return jsonify(resp)

# 6.查询用户
user_info = User.query.filter_by(uid=id).first()
if not user_info: # 校验用户是否获取到数据 没获取到说明用户不存在
    resp["code"] = -1
    resp["msg"] = "指定账号不存在"
    return jsonify(resp)

# 7.判断是不是管理员账户
if user_info and user_info.uid == 1:
    resp["code"] = -1
    resp["msg"] = "该用户是admin账号, 不允许修改编辑 "
    return jsonify(resp)

# 8.根据操作操作类型 改变 账号状态
if act == "remove":
    user_info.status = 0 # 删除状态
elif act == "recover": # 变成不删除状态
    user_info.status = 1

# 9.记录操作时间
user_info.updated_time = getCurrentDate()

# 10.提交保存数据库
db.session.add(user_info)
db.session.commit()

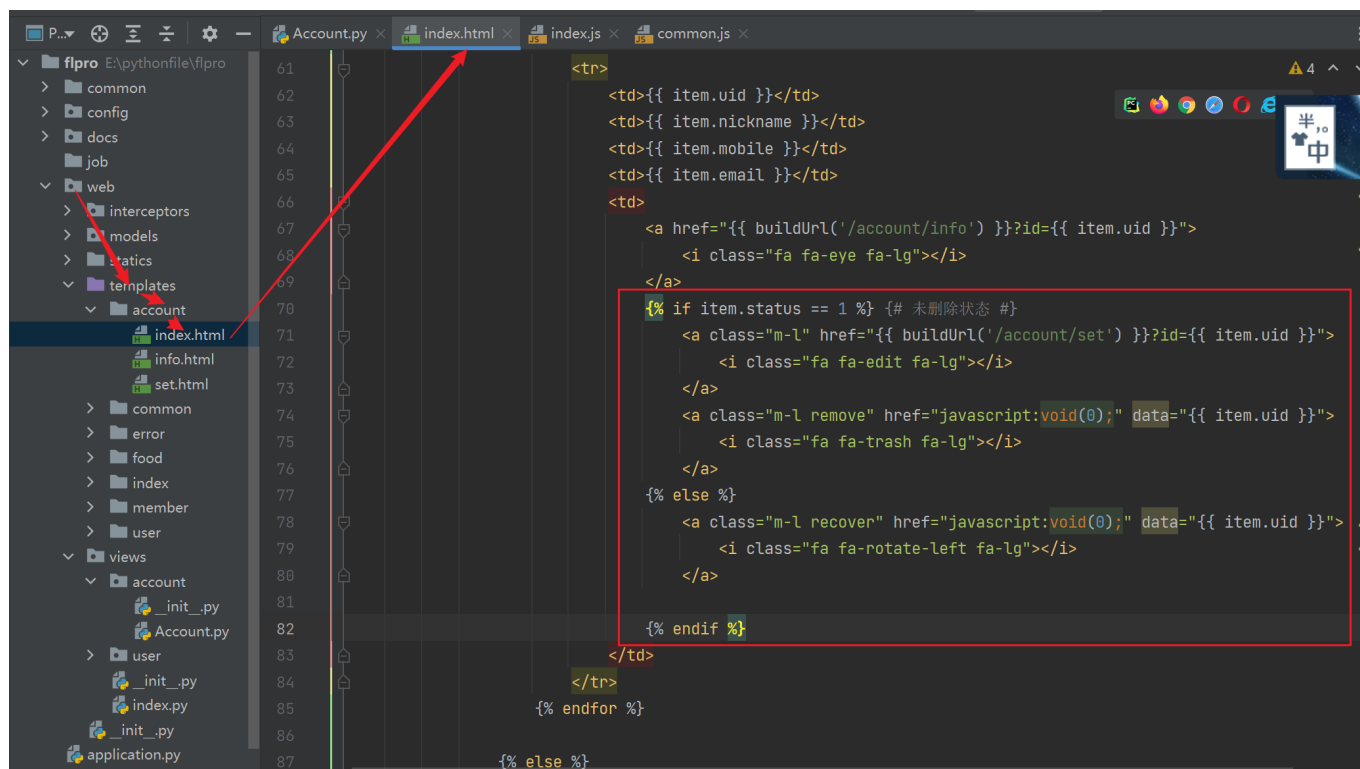
# 响应
return jsonify(resp)

```

```

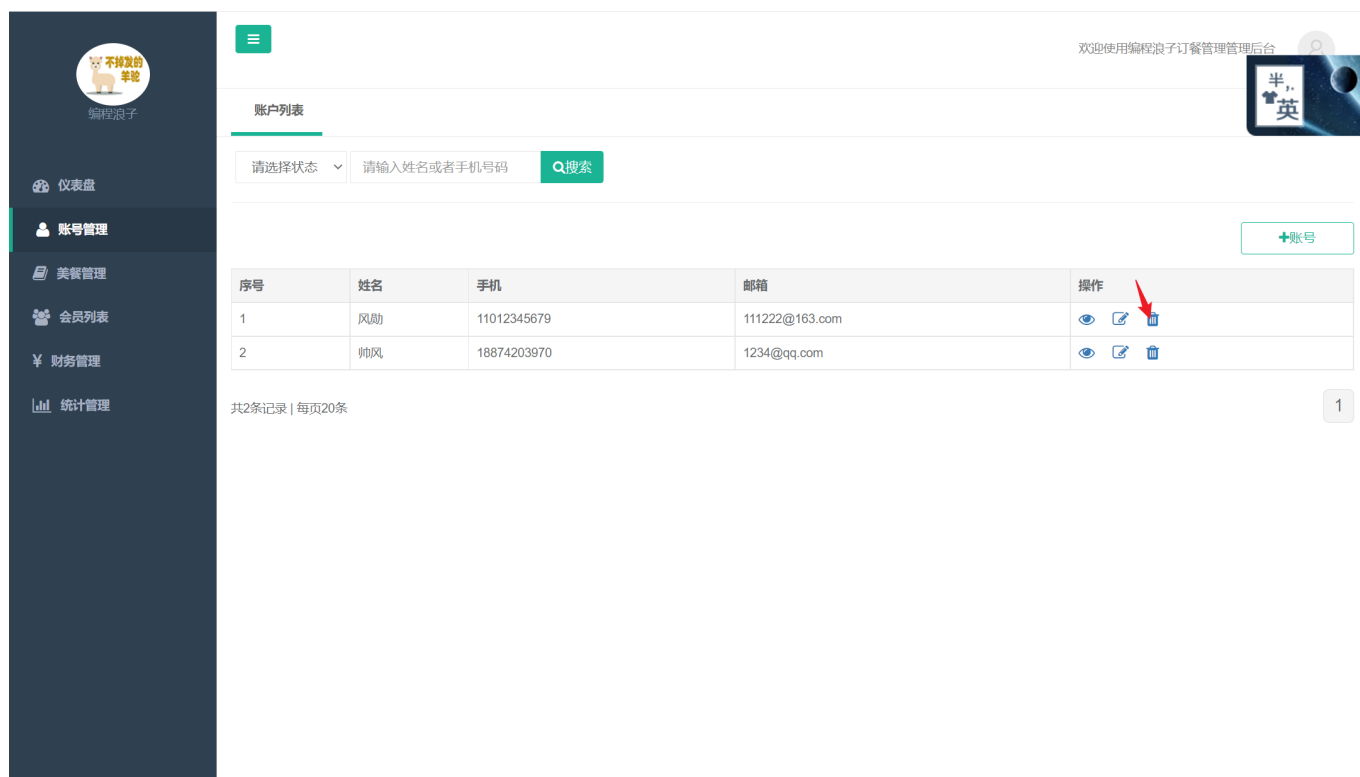
route_account.add_url_rule('/ops', view_func=Ops.as_view('ops'))

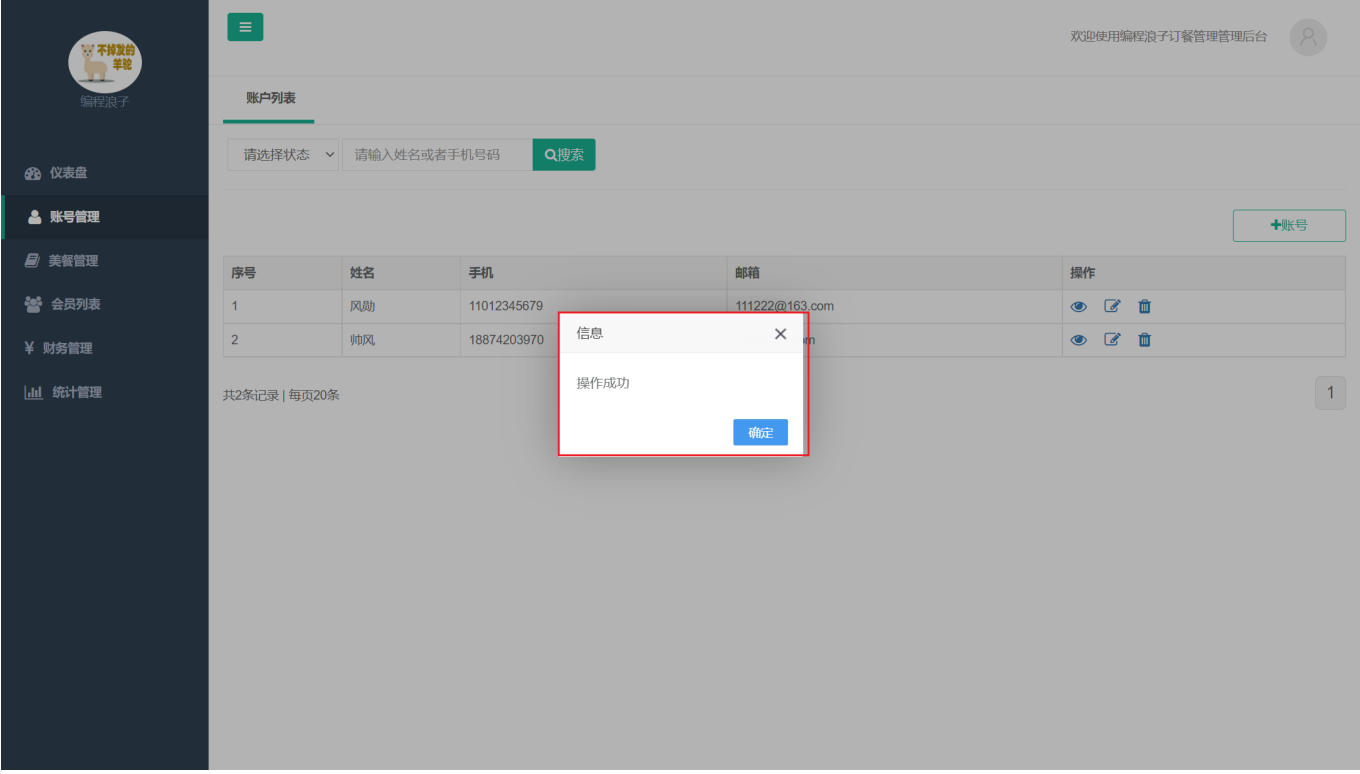
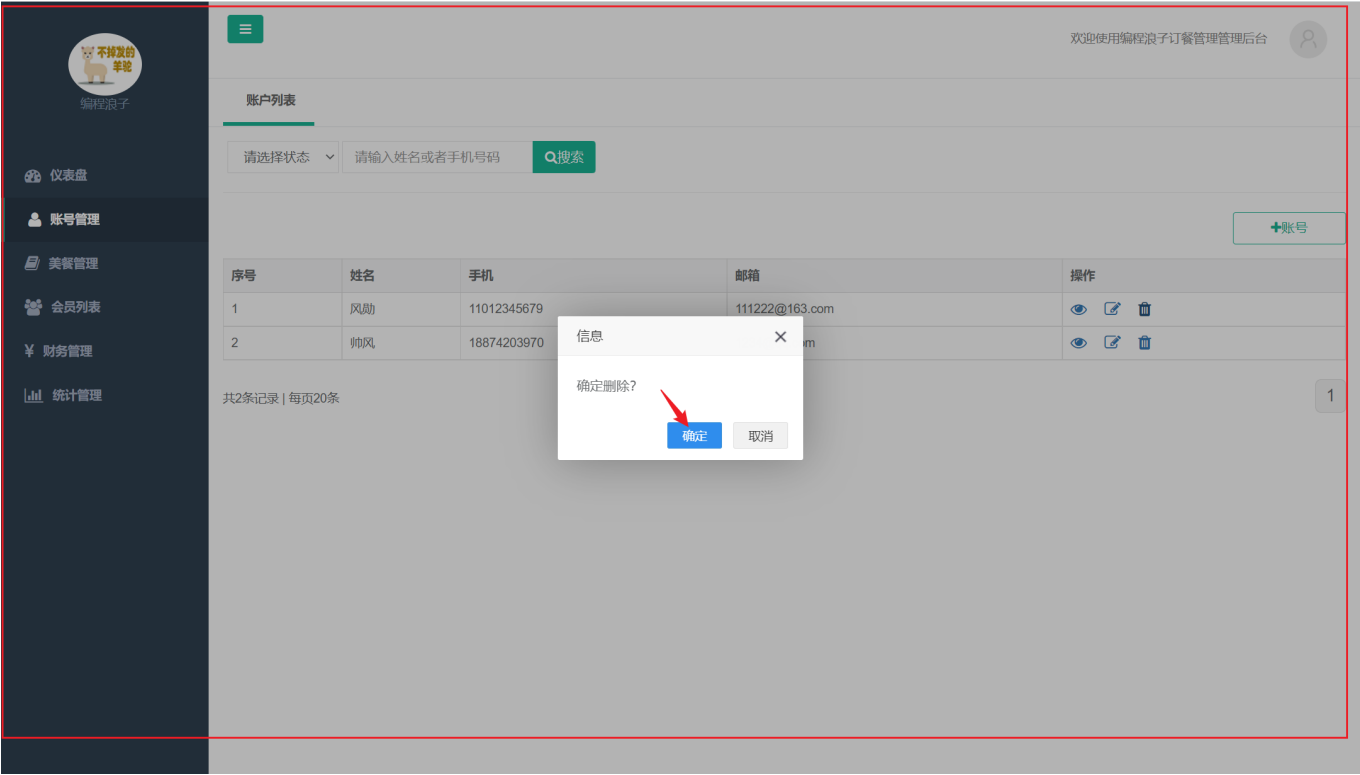
```



启动项目 - 访问 <http://127.0.0.1:8888/account/index>

----删除----





----恢复----

