

1.渲染编辑页

```
Account.py x set.html x
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

class AccountSet(views.MethodView):
    def get(self):
        # 响应
        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'))
```

```
# -*- 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
        current_page = int(req['page']) if 'page' in req else 1
        page_size = int(req['limit']) if 'limit' in req else 10
        offset = (current_page - 1) * page_size
        limit = page_size
        total = query.count()
        user_list = query.offset(offset).limit(limit).all()

        # 封装分页对象
        page_obj = iPagination(total, current_page, page_size)

        # 封装响应的数据
        response_data = {
            'status': 0,
            'msg': '成功',
            'count': total,
            'info': '用户列表',
            'list': user_list,
            'page': page_obj
        }

        return jsonify(response_data)
```

```

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):

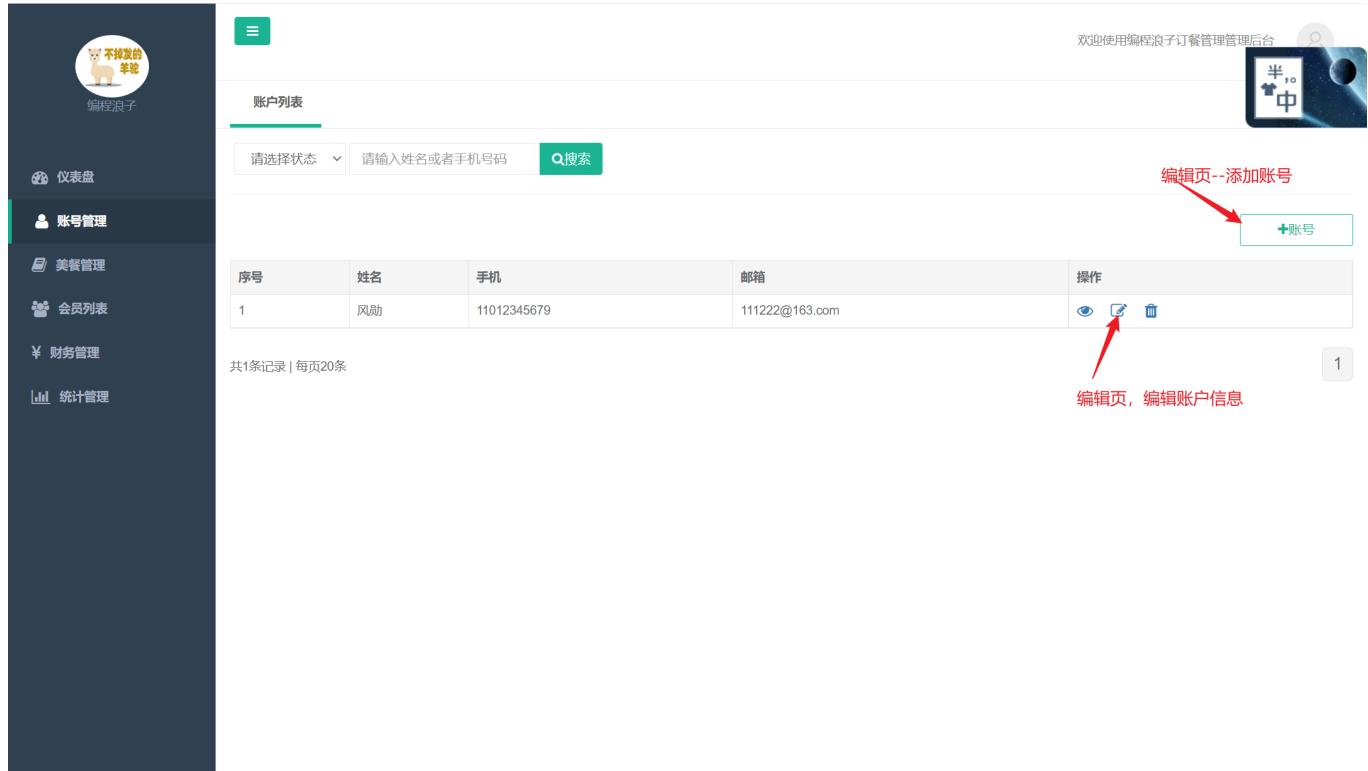
```

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>



The screenshot shows a dark-themed web application interface for account management. On the left is a sidebar with icons for Dashboard, Account Management, Meal Management, Member List, Financial Management, and Statistics Management. The main area has a header with a logo and search bar. Below is a table titled '账户列表' (Account List) with columns: 序号 (ID), 姓名 (Name), 手机 (Phone), 邮箱 (Email), and 操作 (Operations). A single row is shown for '风勋' (Feng Xu) with ID 1, phone 11012345679, email 111222@163.com. The operations column contains three icons: eye, edit, and delete. Red annotations with arrows point to the 'edit' icon in the operations column of the first row and to the '+账号' (Add Account) button in the top right corner of the page.

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

欢迎使用编程浪子订餐管理后台

编辑页--添加账号

+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

The screenshot shows the PyCharm IDE interface. On the left is the project tree with the following structure:

- flpro E:\pythonfile\flpro
 - common
 - config
 - docs
 - job
 - web
 - interceptors
 - models
 - statics
 - templates
 - views
 - account
 - user
 - application.py
 - manager.py
 - www.py
- External Libraries
- Scratches and Consoles

The right pane displays the code for `Account.py`. A red box highlights the class definition and its methods. Red arrows point from the project tree to the `views` and `account` directories.

```
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)  
  
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'))
```

```
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)
```

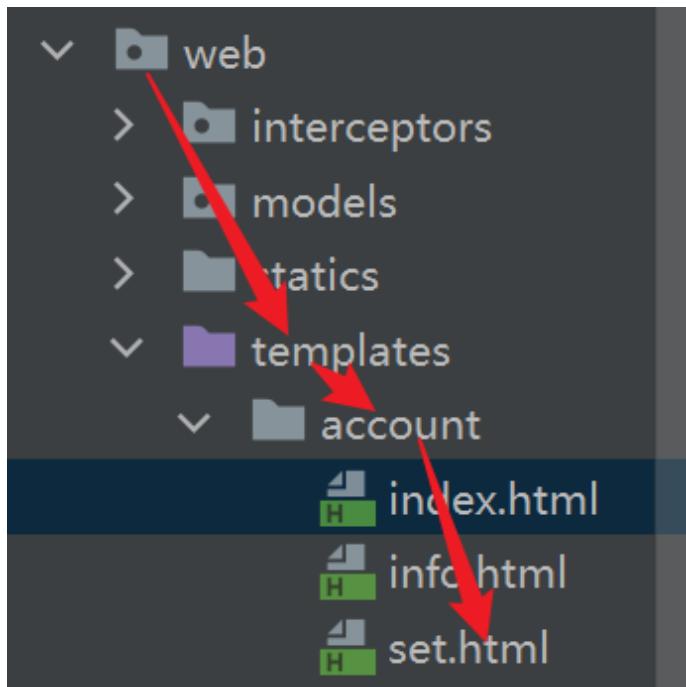
此时前端模板也需要调整

--index.html--

The screenshot shows a code editor interface with two tabs open: `Account.py` and `index.html`. The `index.html` tab is active, displaying an HTML table structure. A red arrow points from the file tree on the left to the `index.html` tab. Another red box highlights a specific line of code in the `index.html` file:

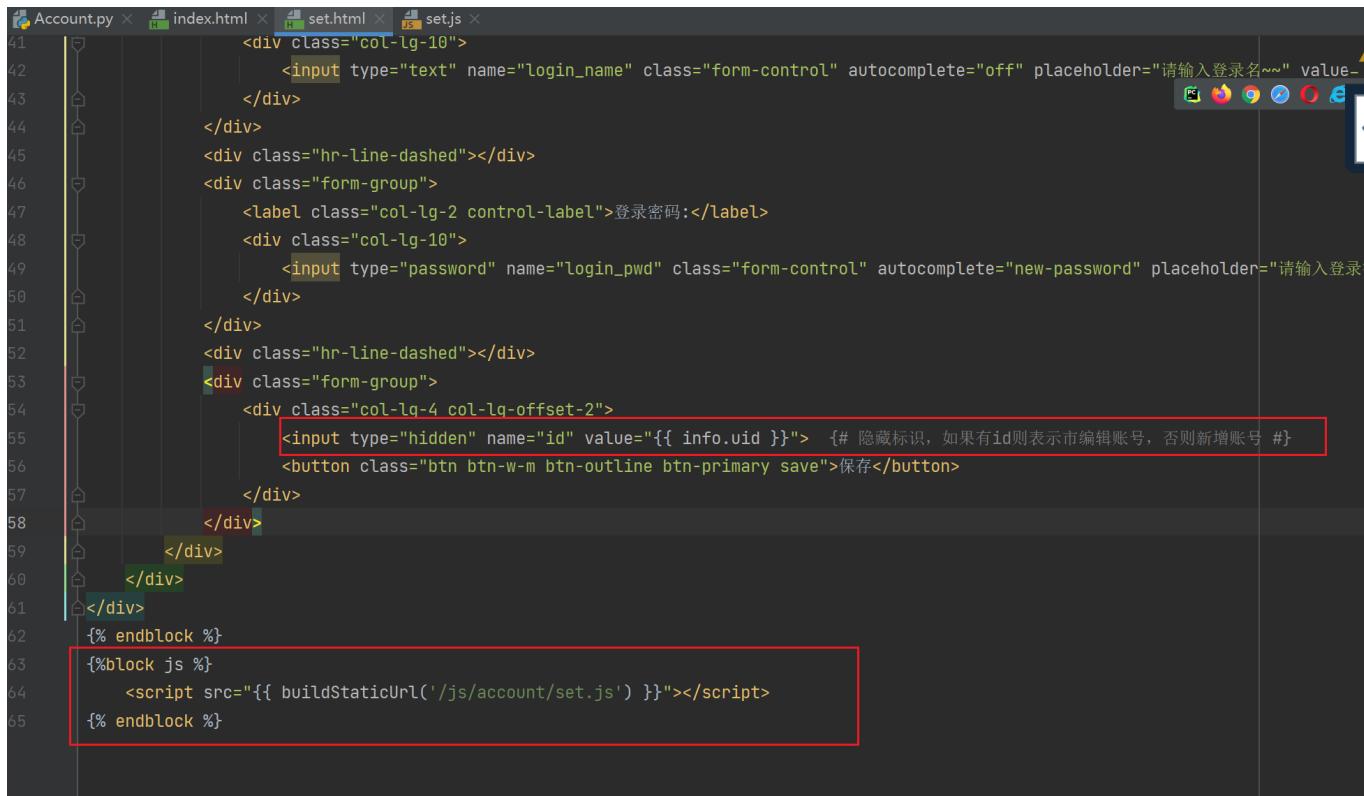
```
% if list %  
    {% for item in list %}  
        <tr>  
            <td>{{ item.uid }}</td>  
            <td>{{ item.nickname }}</td>  
            <td>{{ item.mobile }}</td>  
            <td>{{ item.email }}</td>  
            <td>  
                <a href="{{ buildUrl('/account/info') }}?id={{ item.uid }}">  
                    <i class="fa fa-eye fa-lg"></i>  
                </a>  
                <a class="m-l" href="{{ buildUrl('/account/set') }}?id={{ item.uid }}">  
                    <i class="fa fa-edit fa-lg"></i>  
                </a>  
            </td>  
        </tr>  
    {% endfor %}  
    {% else %}  
        <td colspan="5">暂无数据~</td>  
    {% endif %}  
    </tbody>  
    </table>
```

--set.html--



```
Account.py x index.html x set.html x set.js x
17 <div class="form-group" style="margin-bottom: 10px;">
18   <label class="control-label">姓名:</label>
19   <div class="controls">
20     <input type="text" name="nickname" class="form-control" placeholder="请输入姓名~~" value="{{ info.nickname }}"/>
21   </div>
22 </div>
23 <div class="hr-line-dashed"></div>
24 <div class="form-group" style="margin-bottom: 10px;">
25   <label class="control-label">手机:</label>
26   <div class="controls">
27     <input type="text" name="mobile" class="form-control" placeholder="请输入手机~~" value="{{ info.mobile }}"/>
28   </div>
29 </div>
30 <div class="hr-line-dashed"></div>
31 <div class="form-group" style="margin-bottom: 10px;">
32   <label class="control-label">邮箱:</label>
33   <div class="controls">
34     <input type="text" name="email" class="form-control" placeholder="请输入邮箱~~" value="{{ info.email }}"/>
35   </div>
36 </div>
37 <div class="hr-line-dashed"></div>
38 <div class="form-group" style="margin-bottom: 10px;">
39   <label class="control-label">登录名:</label>
40   <div class="controls">
41     <input type="text" name="login_name" class="form-control" autocomplete="off" placeholder="请输入登录名~~" value="{{ info.login_name }}"/>
42   </div>
43 </div>
```

```
Account.py x index.html x set.html x set.js x
1 <div class="form-group" style="margin-bottom: 10px;">
2   <label class="control-label">邮箱:</label>
3   <div class="controls">
4     <input type="text" name="email" class="form-control" placeholder="请输入邮箱~~" value="{{ info.email }}"/>
5   </div>
6 </div>
7 <div class="form-group" style="margin-bottom: 10px;">
8   <label class="control-label">登录名:</label>
9   <div class="controls">
10    <input type="text" name="login_name" class="form-control" autocomplete="off" placeholder="请输入登录名~~" value="{{ info.login_name }}"/>
11   </div>
12 </div>
13 <div class="form-group" style="margin-bottom: 10px;">
14   <label class="control-label">登录密码:</label>
15   <div class="controls">
16     <input type="password" name="password" class="form-control" autocomplete="new-password" placeholder="请输入登录密码~~" value="{{ info.password }}"/>
17   </div>
18 </div>
19 <div style="text-align: right; margin-top: 10px;">
20   <button type="button" class="btn btn-primary" style="margin-right: 10px;">保存</button>
21   <button type="button" class="btn btn-default">取消</button>
22 </div>
```



```
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 class="hr-line-dashed"></div>
45     <div class="form-group">
46         <label class="col-lg-2 control-label">登录密码:</label>
47         <div class="col-lg-10">
48             <input type="password" name="login_pwd" class="form-control" autocomplete="new-password" placeholder="请输入登录密码" />
49         </div>
50     </div>
51     <div class="hr-line-dashed"></div>
52     <div class="form-group">
53         <div class="col-lg-4 col-lg-offset-2">
54             <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 #}>
55             <button class="btn btn-w-m btn-outline btn-primary save">保存</button>
56         </div>
57     </div>
58     </div>
59     </div>
60 </div>
61 <% endblock %>
62 <% block js %>
63     <script src="{{ buildStaticUrl('/js/account/set.js') }}></script>
64 <% 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 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>
</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>

This screenshot shows the 'Account List' page of a management system. At the top right, there is a welcome message 'Welcome to use the programming lamb's dining management system' and a user icon. On the left, a sidebar menu includes 'Dashboard', 'Account Management' (selected), 'Dining Management', 'Member List', 'Financial Management', and 'Statistics Management'. The main content area has a search bar with dropdowns for 'Status Selection' and 'Search by Name or Phone Number', and a green 'Search' button. A table lists one account:序号 (ID) 1, 姓名 (Name) 风勋, 手机 (Phone) 11012345679, 邮箱 (Email) 111222@163.com. The 'Operation' column for this row contains three icons: a blue eye, a blue edit/pencil, and a blue trash can. A red arrow points to the edit icon. Below the table, it says '1 record |每页20条' (1 record | 20 per page) and a page number '1'.

This screenshot shows the 'Account Settings' page. At the top right, there is a welcome message 'Welcome to use the programming lamb's dining management system' and a user icon. On the left, a sidebar menu includes 'Dashboard', 'Account Management' (selected), 'Dining Management', 'Member List', 'Financial Management', and 'Statistics Management'. The main content area has a title 'Account Settings'. It contains five input fields: 'Name' (风勋), 'Phone' (11012345679), 'Email' (111222@163.com), 'Login Name' (fengxun), and 'Login Password' (*****). A green 'Save' button is at the bottom.

账户列表

请选择状态 ▾ 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	

共1条记录 | 每页20条 1

+账号

账户设置

姓名:

手机:

邮箱:

登录名:

登录密码:

保存

3. 编辑与新建实现 -- js分析

The screenshot shows a code editor with two tabs: 'Account.py' and 'set.js'. The 'set.js' tab is active. A red arrow points from the file tree on the left to the 'set.js' code. The code is as follows:

```
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       }                                判断是否处于提交状态
13
14       var nickname_target = $(".wrap_account_set input[name=nickname]");
15       var nickname = nickname_target.val();
16
17       var mobile_target = $(".wrap_account_set input[name=mobile]");
18       var mobile = mobile_target.val();      获取相关提交需要的数据
19
20       var email_target = $(".wrap_account_set input[name=email]");
21       var email = email_target.val();
22
23       var login_name_target = $(".wrap_account_set input[name=login_name]");
24       var login_name = login_name_target.val();
25
26       var login_pwd_target = $(".wrap_account_set input[name=login_pwd]");
27       var login_pwd = login_pwd_target.val();
28
29       if( nickname.length < 1 ){
30         common_ops.tip( msg: "请输入符合规范的姓名~~", nickname_target );
```

The screenshot shows a code editor with two tabs: 'Account.py' and 'set.js'. The 'set.js' tab is active. A large red box highlights a block of code. The code is as follows:

```
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_one: eventBind() → callback for $("wrap_account_set .save").click()
```

A red box highlights the entire block of validation logic (lines 8-34). A red annotation '校验数据是否符合需求' (Validate data to meet requirements) is placed above the validation logic.

```
52     }
53     btn_target.addClass( value: "disabled" );
54     var data = {
55       nickname: nickname,
56       mobile: mobile,
57       email: email,
58       login_name: login_name,
59       login_pwd: login_pwd,
60       id:$(".wrap_account_set input[name=id]").val()
61   };
62
63     $.ajax( url: {
64       url:common_ops.buildUrl( path: "/account/set" ),
65       type:'POST',
66       data:data,
67       dataType:'json',
68       success:function( res ){
69         btn_target.removeClass( value: "disabled" );
70         var callback = null;
71         if( res.code == 200 ){
72           callback = function(){
73             window.location.href = common_ops.buildUrl( path: "/account/index" );
74           }
75         }
76         common_ops.alert( res.msg,callback ); /*后端返回错误提示*/
77       }
78     });

```

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

```
Account.py x Helper.py x UserService.py x
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144

class AccountSet(views.MethodView):
    def get(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>

The screenshot shows a user management application. At the top right, there is a welcome message "欢迎使用编程浪子订餐管理后台" and a user profile icon. Below the header, there is a search bar with dropdown filters for status and name, and a search button. A red arrow points to a green button labeled "+账号" (New Account) located at the bottom right of the page. The main area displays a table of users with columns: 序号 (Index), 姓名 (Name), 手机 (Phone), 邮箱 (Email), and 操作 (Operations). One row is shown with the following data:

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

At the bottom left, it says "共1条记录 | 每页20条". On the far right, there is a small page number "1".

账号设置

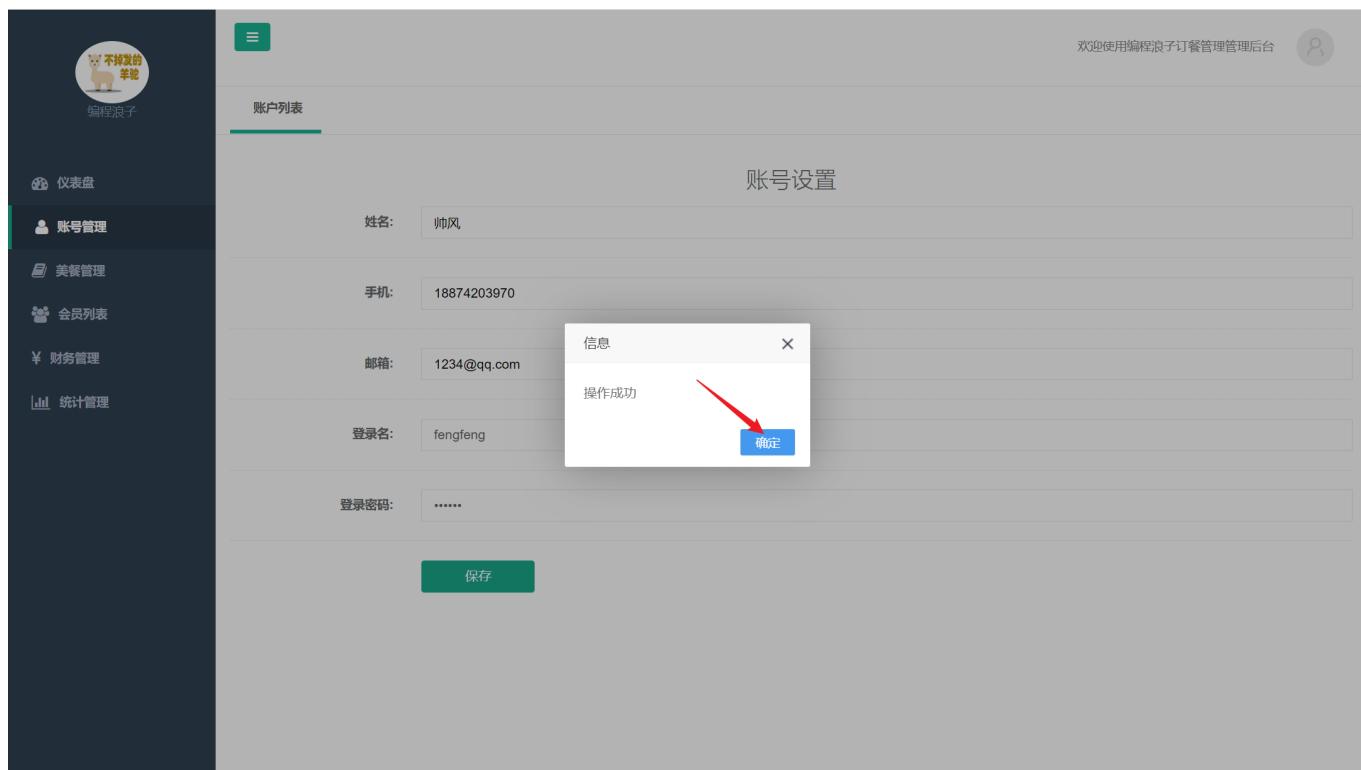
姓名: 帅风

手机: 18874203970

邮箱: 1234@qq.com

登录名: fengfeng

登录密码: 123456

保存

欢迎使用编程浪子订餐管理后台

账户列表

账号设置

姓名: 帅风

手机: 18874203970

邮箱: 1234@qq.com

登录名: fengfeng

登录密码:

信息

操作成功

确定

保存



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	
2	帅风	18874203970	1234@qq.com	

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

点击后



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

共2条记录 | 每页20条

```
Account.js x index.js x
1;
2 var account_index_ops = {
3     init:function(){
4         this.eventBind();
5     },
6     eventBind:function(){
7         var that = this;
8         $(" .wrap_search .search").click(function(){
9             $(" .wrap_search").submit(); /*搜索, 传递参数mix_kw混合查询关键字*/
10        });
11
12         $(".remove").click( function(){
13             that.ops( act: "remove",$(this).attr( name: "data") );
14         });
15
16         $(".recover").click( function(){
17             that.ops( act: "recover",$(this).attr( name: "data") );
18         });
19
20         ops:function( act,id ){
21             var callback = {
22                 'ok':function(){
23                     $.ajax( url: {
24                         url:common_ops.buildUrl( path: "/account/ops" ),
25                         type:'POST',
26                         data:{ act:act,
27                               id:id
28                           },
29                         dataType:'json',
30                         success:function( res ){
31                             var callback = null;
32                             if( res.code == 200 ){
33                                 callback = function(){
34                                     window.location.href = window.location.href;
35                                 }
36                             }
37                             common_ops.alert( res.msg,callback );
38                         }
39                     );
40                 },
41                 'cancel':null
42             };
43             /*同意进行删除前, 进行一个提示, 否则删除很危险, 在common.js中定义了callback方法*/
44             common_ops.confirm( ( act == "remove" ? "确定删除? ":"确定恢复? " ), callback );
45         };
46     };
47 }
```

```
ops:function( act,id ){
    var callback = {
        'ok':function(){
            $.ajax( url: {
                url:common_ops.buildUrl( path: "/account/ops" ),
                type:'POST',
                data:{ act:act,
                      id:id
                    },
                dataType:'json',
                success:function( res ){
                    var callback = null;
                    if( res.code == 200 ){
                        callback = function(){
                            window.location.href = window.location.href;
                        }
                    }
                    common_ops.alert( res.msg,callback );
                }
            );
        },
        'cancel':null
    };
    /*同意进行删除前, 进行一个提示, 否则删除很危险, 在common.js中定义了callback方法*/
    common_ops.confirm( ( act == "remove" ? "确定删除? ":"确定恢复? " ), callback );
};
```

7. 删除与恢复账号

```

flpro E:\pythonfile\flpro
  common
  config
  docs
  job
  web
    interceptors
    models
    statics
    templates
      account
        index.html
        info.html
        set.html
    common
    error
    food
    index
    member
    user
  views
    account
      __init__.py
      Account.py
    user
      __init__.py
      index.py
    __init__.py
  application.py

Account.py x index.html x index.js x
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104

```

index.html content:

```

    ...
    <!-- 分页代码已被封装到统一模板文件中-->
    <div class="row">#
      <div class="col-lg-12">#
        <span class="pagination_count" style="line-height: 40px;">共1条记录 | 每页50条</span>#
        <ul class="pagination pagination-lg pull-right" style="margin: 0 0 ;">#
          <li class="active"><a href="javascript:void(0);">1</a></li>#
        </ul>#
      </div>#
    </div>
  </div>
  {% endblock %}

  {%block js %}
    <script src="{{ buildStaticUrl('/js/account/index.js') }}></script>
  {% endblock %}

```

```

flpro E:\pythonfile\flpro
  common
  config
  docs
  job
  web
    interceptors
    models
    statics
    templates
    views
      account
        __init__.py
        Account.py
    user
      __init__.py
      index.py
    __init__.py
  application.py
  manager.py
  www.py

  External Libraries
  Scratches and Consoles

Account.py x index.js x
19
67
68
69
99
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271

```

Account.py content:

```

class Index(views.MethodView):...
class Info(views.MethodView):...
class AccountSet(views.MethodView):...
# 删除和恢复账号
class Ops(views.MethodView):
    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 携带的数据对象
        req = request.values

        # 3.获取数据 act -- 操作类型    id --- 操作用户id
        act = req["act"] if "act" in req else ""
        id = req["id"] if "id" in req else 0

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

```

```

# 删除和恢复账号
class Ops(views.MethodView):
    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 携带的数据对象
        req = request.values

        # 3.获取数据 act -- 操作类型    id --- 操作用户id
        act = req["act"] if "act" in req else ""
        id = req["id"] if "id" in req else 0

```

```
# 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: # 校验用户是否获取导数据 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'))
```

```

<tr>
  <td>{{ item.uid }}</td>
  <td>{{ item.nickname }}</td>
  <td>{{ item.mobile }}</td>
  <td>{{ item.email }}</td>
  <td>
    <a href="{{ buildUrl('/account/info') }}?id={{ item.uid }}">
      <i class="fa fa-eye fa-lg"></i>
    </a>
    {% if item.status == 1 %} {# 未删除状态 #}
      <a class="m-l" href="{{ buildUrl('/account/set') }}?id={{ item.uid }}">
        <i class="fa fa-edit fa-lg"></i>
      </a>
      <a class="m-l remove" href="javascript:void(0); data="{{ item.uid }}">
        <i class="fa fa-trash fa-lg"></i>
      </a>
    {% else %}
      <a class="m-l recover" href="javascript:void(0); data="{{ item.uid }}">
        <i class="fa fa-rotate-left fa-lg"></i>
      </a>
    {% endif %}
  </td>
</tr>
{% endfor %}
{% else %}

```

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

----删除----

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

This screenshot shows the 'Account List' page of a management system. On the left is a dark sidebar with icons for Dashboard, Account Management, Meal Management, Member List, Financial Management, and Statistics Management. The main area has a light gray background. At the top right, it says 'Welcome to use the programming lamb's dining management后台' and has a user profile icon. Below that is a search bar with dropdowns for status and name, and a search button. A green button labeled '+Account' is on the right. The table below has columns for序号 (Index), 姓名 (Name), 手机 (Phone), 邮箱 (Email), and 操作 (Operations). Two rows are shown: one for '风勋' (Phone: 11012345679, Email: 111222@163.com) and another for '帅风' (Phone: 18874203970, Email: 18874203970). A modal dialog box is centered over the second row, asking '确定删除?' (Delete confirmed?). It has a red arrow pointing to the blue '确定' (Confirm) button.

This screenshot shows the same 'Account List' page after a deletion. The modal dialog from the previous screen is still open, but its content has changed to '操作成功' (Operation successful). The red box and arrow from the previous screenshot are present here, highlighting the '确定' (Confirm) button in the dialog.

----恢复----

编程浪子

欢迎使用编程浪子订餐管理后台

账户列表

请选择状态 搜索

序号	姓名	手机	邮箱	操作
1	风勋	11012345679	111222@163.com	
2	帅风	18874203970		

共2条记录 | 每页20条

+账号

信息

确定恢复?

确定 取消

编程浪子

欢迎使用编程浪子订餐管理后台

账户列表

请选择状态 搜索

序号	姓名	手机	邮箱	操作
1	风勋	11012345679	111222@163.com	
2	帅风	18874203970		

共2条记录 | 每页20条

+账号

信息

操作成功

确定