

# 1. 基本分析

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	风励	11012345679	111222@163.com	
2	帅风	18874203970	1234@qq.com	

账号

统计管理

共2条记录 | 每页20条

1

注意：这里提交请同学们一定要熟悉 表单提交 数据是如何传递的。

当触发提交submit事件后，input相应的数据会在提交时整合到查询字符串内

我们可以点击上述搜索后观看地址栏的url地址变化

127.0.0.1:8888/account/index?status=-1&mix\_kw=&p=1

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	
2	帅风	18874203970	1234@qq.com	

共2条记录 | 每页20条

我们观察下html看下表单是什么情况

web/templates/account/index.html

```

11 </div>
12 </div>
13 </div>
14 <div class="row">
15 <div class="col-lg-12">
16 <form class="form-inline wrap_search"> {# 默认get请求，并且请求的是当前页面的路由 #-}
17 <div class="row m-t p-w-m">
18 <div class="form-group">
19 <select name="status" class="form-control inline">
20 <option value="-1">请选择状态</option>
21 <option value="1">正常</option>
22 <option value="0">已删除</option>
23 </select>
24 </div>
25 <div class="form-group">
26 <div class="input-group">
27 <input type="text" name="mix_kw" placeholder="请输入姓名或者手机号码" class="form-control" value="" />
28 <input type="hidden" name="p" value="1" />
29 <span class="input-group-btn">
30 <button type="button" class="btn btn-primary search">
31 <i class="fa fa-search"></i> 搜索
32 </button>
33 </span>
34 </div>
35 </div>
36 </div>
37 </div>
38 <hr>

```

结合到我们的前面的点击后搜索按钮后，url地址变化基本上就能很好的去解释form表单数据提交了

同学们观看html后，发现我们搜索按钮是如何提交的呢？

```
Account.py x index.html x index.js x
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

按钮 -- 并非提交按钮 咋提交的呢?

既然这里没有提交按钮submit，那么说明这里采用js触发的，所以我们去观察js

web/statics/js/account/index.js

```
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:{  
27                         act:act,  
28                         id:id  
29                     }  
30                 }  
31             }  
32         };  
33     }  
34 }  
35 
```

基本的前端提交解决了，那么接下来我们就根据提交数据去书写搜索逻辑

[http://127.0.0.1:8888/account/index?status=-1&mix\\_kw=&p=1](http://127.0.0.1:8888/account/index?status=-1&mix_kw=&p=1)

status --- 用户状态 --- 结合html可知： -1--代表全部状态 0--代表已删除 1--代表正常

mix\_kw --- 搜索数据

p -- 代表页码 --- 当然这里无关紧要了

## 2. 后端实现

The screenshot shows the PyCharm IDE interface. On the left is the Project Explorer, displaying the file structure of the 'flipro' project. The 'index' module is selected. In the center is the code editor for the 'Index.py' file under the 'views/account' package. A red arrow points from the 'index.html' tab in the Project Explorer to the code editor. Another red arrow points from the 'Account.py' file in the Project Explorer to the 'Index' class definition in the code editor. The code editor contains Python code for handling user search requests.

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

class Index(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?

        # 搜索功能
        if "mix_kw" in req: # 请求数据中存在 mix_kw 说明发起了搜索功能

            # 建立 查询逻辑或对象
            # from sqlalchemy import or_
            # 逻辑或 --- or_
```

```
class Index(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?

```
# 搜索功能
if "mix_kw" in req: # 请求数据中存在 mix_kw 说明发起了搜索功能

    # 建立 查询逻辑或对象
    # from sqlalchemy import or_
    # 逻辑或 --- or_
    # nickname 或者 mobile 数据是 mix_kw数据 那么就代表 为目标需求数据
    rule = or_(User.nickname.ilike(f"%{req['mix_kw']}%"),
               User.mobile.ilike(f"%{req['mix_kw']}%"))

    # 用户数据对象 查询筛选 数据
    query = query.filter(rule)

# 请求数据中存在 status --- status代表状态 ; -1 -- 所有符合条件 , 1 -- 正常 , 0 -- 已删除
if "status" in req and int(req["status"]) > -1:
    # status 为 0 或者 1 就 筛选对应状态的数据
    query = query.filter(User.status == int(req["status"]))

# 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, # 分页数据
    "search_on": req, # 搜索参数
    "status_mapping": {"1": "正常", "0": "已删除"} # 状态映射数据
}

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

```
# 搜索功能
if "mix_kw" in req: # 请求数据中存在 mix_kw 说明发起了搜索功能

    # 建立 查询逻辑或对象
    # from sqlalchemy import or_
    # 逻辑或 --- or_
    # nickname 或者 mobile 数据是 mix_kw数据 那么就代表 为目标需求数据
    rule = or_(User.nickname.ilike(f"%{req['mix_kw']}%"),
               User.mobile.ilike(f"%{req['mix_kw']}%"))

    # 用户数据对象 查询筛选 数据
    query = query.filter(rule)

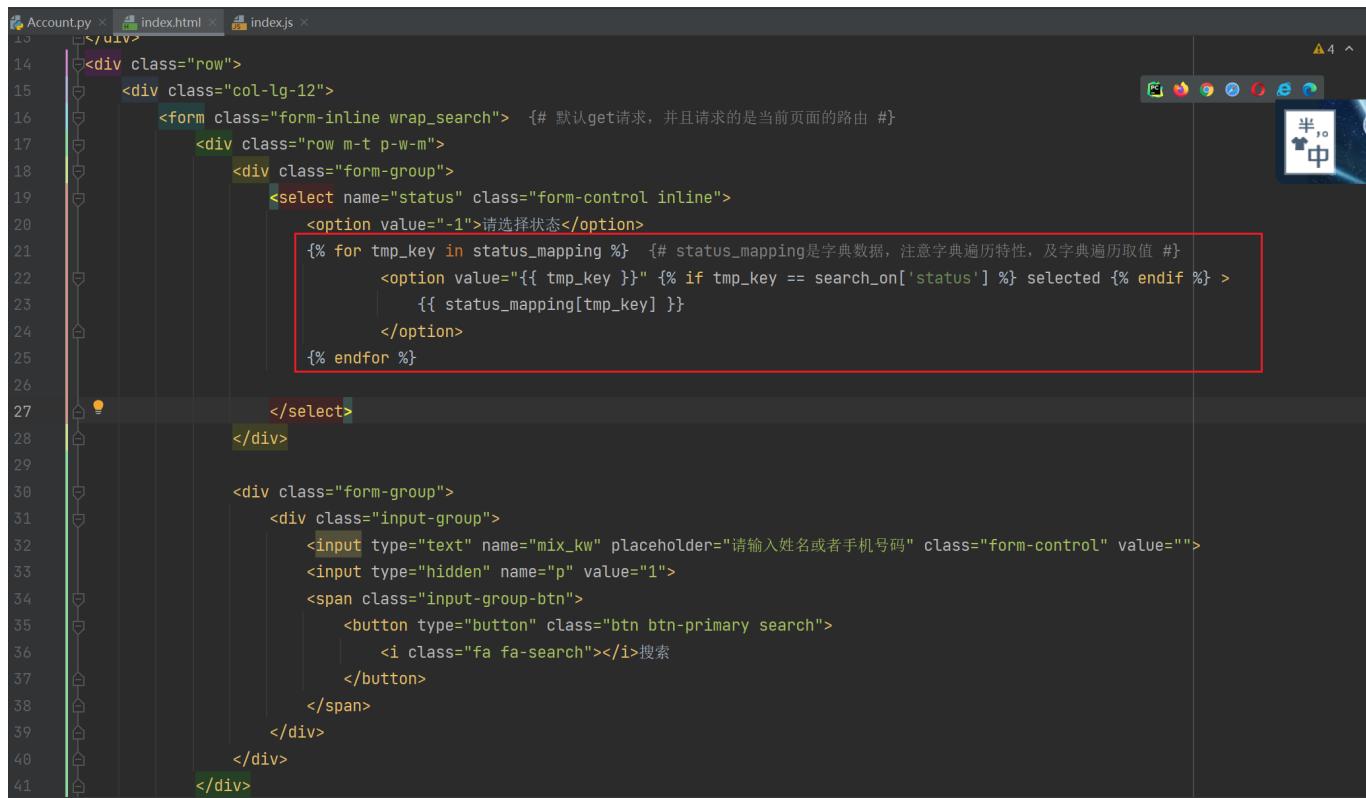
# 请求数据中存在 status --- status代表状态 ; -1 -- 所有符合条件 , 1 -- 正常 , 0 -- 已删除
if "status" in req and int(req["status"]) > -1:
    # status 为 0 或者 1 就 筛选对应状态的数据
    query = query.filter(User.status == int(req["status"]))


```

```
# 9. 构建响应数据
context = {
    "list": user_all[offset:limit], # 展示数据信息列表
    "pages": pages, # 分页数据
    "search_on": req, # 搜索参数
    "status_mapping": {"1": "正常", "0": "已删除"} # 状态映射数据
}
```

==模板调整==

web/templates/account/index.html

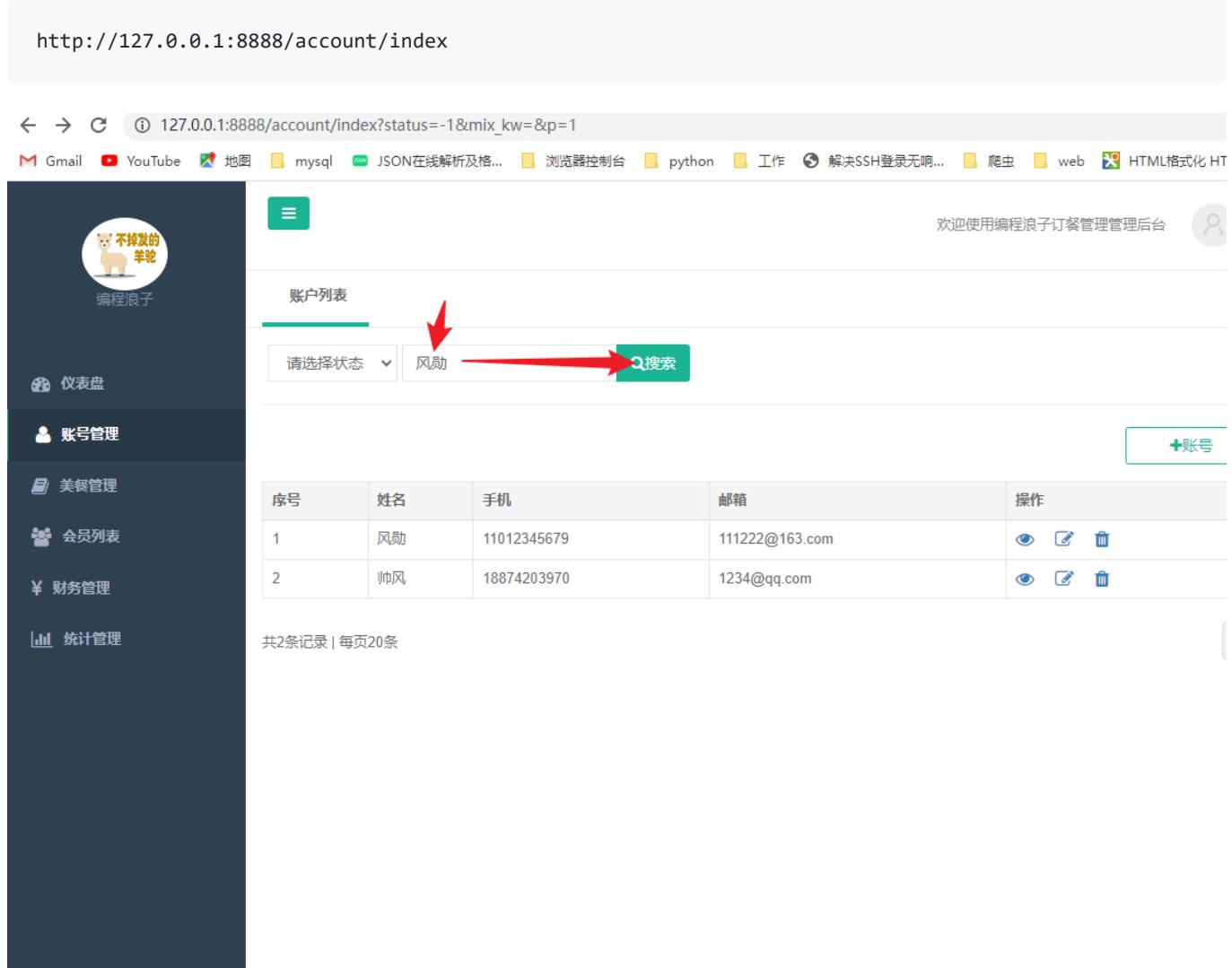


```
14 <div class="row">
15   <div class="col-lg-12">
16     <form class="form-inline wrap_search"> {# 默认get请求，并且请求的是当前页面的路由 #}
17       <div class="row m-t p-w-m">
18         <div class="form-group">
19           <select name="status" class="form-control inline">
20             <option value="-1">请选择状态</option>
21             {% for tmp_key in status_mapping %} {# status_mapping是字典数据，注意字典遍历特性，及字典遍历取值 #}
22               <option value="{{ tmp_key }}"{% if tmp_key == search_on['status'] %} selected {% endif %}>
23                 {{ status_mapping[tmp_key] }}
24               </option>
25             {% endfor %}
26
27           </select>
28         </div>
29
30         <div class="form-group">
31           <div class="input-group">
32             <input type="text" name="mix_kw" placeholder="请输入姓名或者手机号码" class="form-control" value="">
33             <input type="hidden" name="p" value="1">
34             <span class="input-group-btn">
35               <button type="button" class="btn btn-primary search">
36                 <i class="fa fa-search"></i>搜索
37               </button>
38             </span>
39           </div>
40         </div>
41       </div>

```

### 3.启动访问测试

http://127.0.0.1:8888/account/index



左侧导航栏：

- 不掉发的羊驼
- 编程浪子
- 仪表盘
- 账号管理
- 美餐管理
- 会员列表
- 财务管理
- 统计管理

右侧内容区：

欢迎使用编程浪子订餐管理后台

账户列表

请选择状态 ▾ 风勋 

搜索

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

- → C ⓘ 127.0.0.1:8888/account/index?status=-1&mix\_kw=风勋&p=1

Gmail YouTube 地图 mysql JSON在线解析及格... 浏览器控制台 python 工作 解决SSH登录无响应 爬虫 web HTML格式化 HTM

不掉发的  
羊驼 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 ▼ 请输入姓名或者手机号码

+账号

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

仪表盘 账号管理 美餐管理 会员列表 财务管理 统计管理