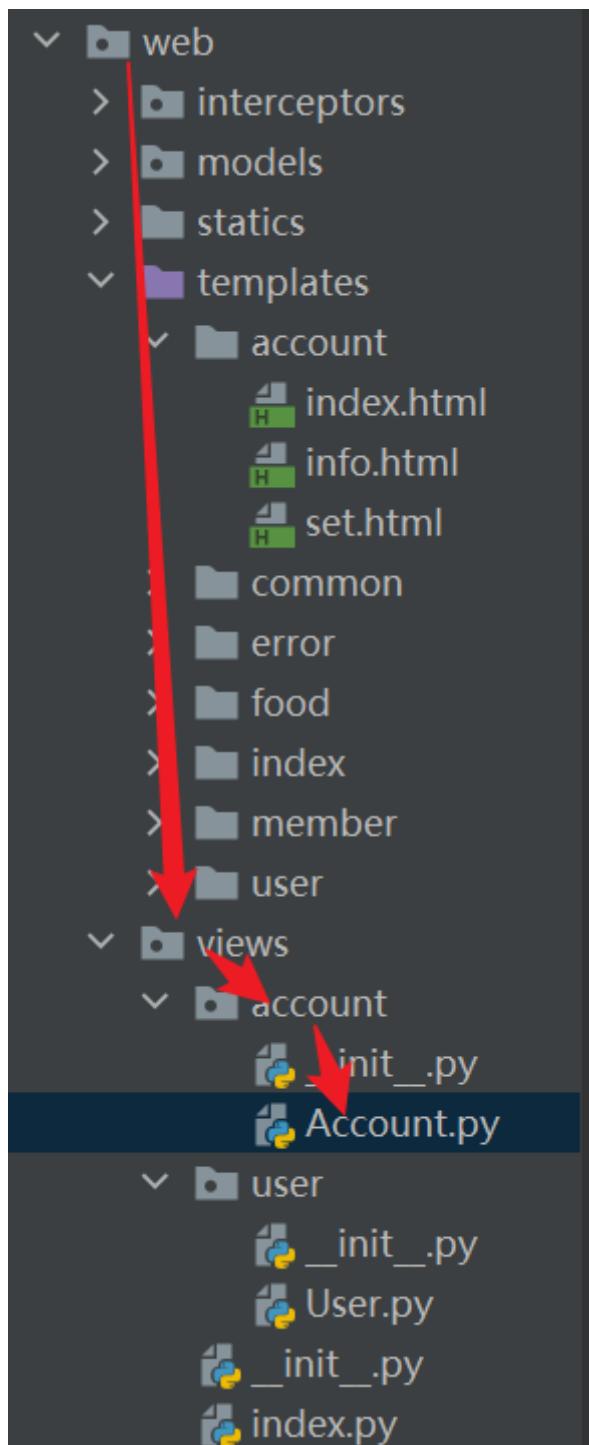


1. 渲染账号管理页面

建立用户列表页视图模块

web/views目录下创建 account目录，account目录下创建Account.py文件



建立视图

```
# -*- 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

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

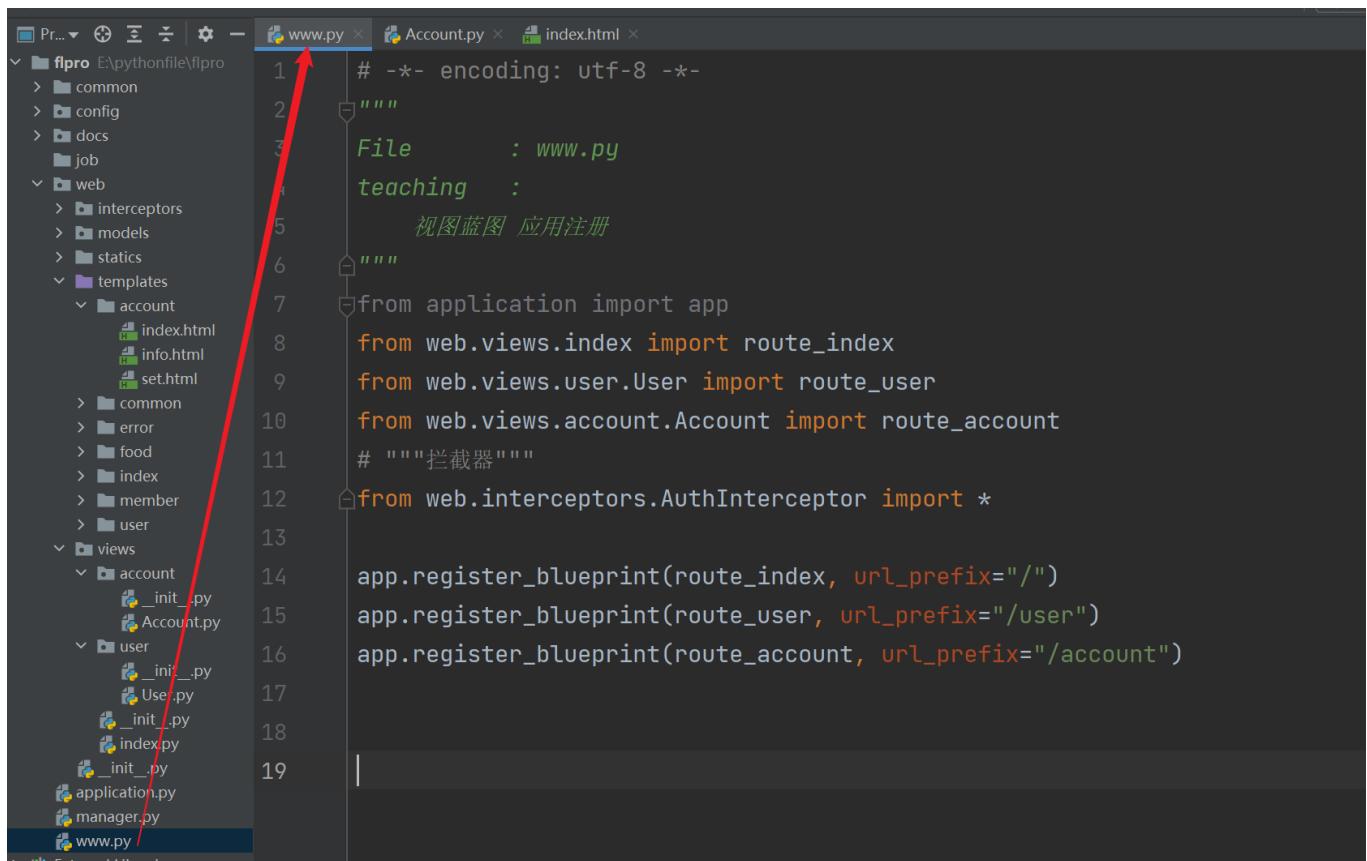
class Index(views.MethodView):

    def get(self):
        return render_template('account/index.html')

route_account.add_url_rule('/index', view_func=Index.as_view('index'))

```

注册蓝图



```

# -*- encoding: utf-8 -*-
"""
File      : www.py
teaching   :
    视图蓝图 应用注册
"""

from application import app
from web.views.index import route_index
from web.views.user.User import route_user
from web.views.account.Account import route_account
# """拦截器"""
from web.interceptors.AuthInterceptor import *

app.register_blueprint(route_index, url_prefix="/")
app.register_blueprint(route_user, url_prefix="/user")
app.register_blueprint(route_account, url_prefix="/account")

```

```

# -*- encoding: utf-8 -*-
"""
File      : www.py
teaching   :

```

视图蓝图 应用注册

```
from application import app
from web.views.index import route_index
from web.views.user.User import route_user
from web.views.account.Account import route_account
# """拦截器"""
from web.interceptors.AuthInterceptor import *

app.register_blueprint(route_index, url_prefix="/")
app.register_blueprint(route_user, url_prefix="/user")
app.register_blueprint(route_account, url_prefix="/account")
```

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

The screenshot shows a user management interface. On the left is a sidebar with icons for Dashboard, Account Management, Meal Management, Member Management, Financial Management, and Statistics Management. The main area has a header '账户列表' (User List) with search filters for status and name. A table lists one user:序号 (ID), 姓名 (Name), 手机 (Phone), 邮箱 (Email), and 操作 (Actions). The table includes a '添加账号' (Add Account) button at the top right.

序号	姓名	手机	邮箱	操作
1	编程浪子2	11012031511	apanly@163.com	

2.分页器

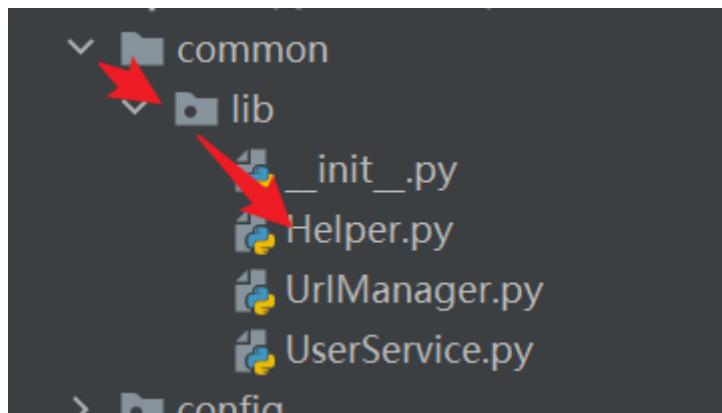
This screenshot shows a user list page with a table and pagination controls. The table has columns for Name, Phone, Email, and Operations. Below the table, it says '共1条记录 | 每页50条'. To the right, there's a '显示页码栏数量' (Show Page Number) button and a page number '1'.

姓名	手机	邮箱	操作
风勋	11012345679	111222@163.com	

显示页码栏数量 如下图 共计九个
当然下图仅做参考



common/lib下创建Helper.py文件



```
# -*- encoding: utf-8 -*-
"""
File      : Helper.py
teaching   :

"""
import math

def iPagination(params):
    """
    :param params:
    {
        'total':xxxx,      # 数据总数
        'page_size':xxxx,  # 每页数据数量
        'page':xxxx,       # 当前所处页
        'display':xxxx,   # 显示页码栏数量
        'url':xxx,        # 路由
    }
    :return:
    """
    # 构造分页器对象
    ret = {
        "is_prev": 1,     # 是否有上一页
        "is_next": 1,    # 是否有下一页
        "from": 0,       # 页码起始位置
        "end": 0,        # 页码终止位置
        "current": 0,    # 当前页
        "total_pages": 0, # 全部页码数
        "page_size": 0,   # 每一页的数量
        "total": 0,       # 全部数量
        "url": params['url'] # 路由
    }
```

```
}

# 获取数据总数
total = int(params['total'])

# 获取每页数据数量
page_size = int(params['page_size'])

# 当前所处页
page = int(params['page'])

# 显示页码栏数量
display = int(params['display'])

# 向上取整 math.ceil
# 数据总数 除以 每页数据数量 = 页码总数
total_pages = int(math.ceil(total / page_size))

# 如果 total_pages大于0 说明存在多页数据 反之就只有一页数据
total_pages = total_pages if total_pages > 0 else 1

# 当前所处页 <= 1 表示没有上一页
if page <= 1:
    ret['is_prev'] = 0 # 表示没有上一页

# 当前所处页 >= total_pages(页码总数) # 表示没有下一页
if page >= total_pages:
    ret['is_next'] = 0 # 表示没有下一页

# 将显示页 整除 2 --- 结果向上取整
# 显示页码栏数量 对半 --- 前面展示哪部分数据， 后面展示哪一部分数据
semi = int(math.ceil(display / 2))

# 当前所处页减去 显示页码栏数量对半 --- 得到的是展示栏前一半 需要展示 起始参数
if page - semi > 0:
    # 如果大于0, 说明我们的 起始栏数据 能正常填充完整, 反之不能
    ret['from'] = page - semi
else:
    ret['from'] = 1

# 当前所处页加上 显示页码栏数量对半 --- 得到的是展示栏后一半 需要展示 结束参数
if page + semi <= total_pages:
    # 如果小于0总页数, 说明我们的结束栏数据能正常填充完整, 反之不能
    ret['end'] = page + semi
else:
    ret['end'] = total_pages

ret['current'] = page # 当前页
ret['total_pages'] = total_pages # 总页数
ret['page_size'] = page_size # 每页数据数量
ret['total'] = total # 数据总数
# 生成数据栏 迭代对象 --- 起始,结束
ret['range'] = range(ret['from'], ret['end'] + 1)
return ret
```

3.配置页码参考配置

```
config/base_setting.py
```

```
# -*- encoding: utf-8 -*-
"""
File      : config/base_setting.py
Time      : 2022/11/18 14:34
teaching   :
    flask配置
"""

SERVER_PORT = 8888
DEBUG = False
SQLALCHEMY_ECHO = False
SECRET_KEY = 'fljgklajdglkajgkljaglka'
# 解决中文在浏览器中显示的问题
JSON_AS_ASCII = False

AUTH_COOKIE_NAME = "food"

# 忽略登录验证的 路由
IGNORE_URLS = [
    "^/user/login", # 登录路由
]
# 忽略登录验证的 资源加载路由
IGNORE_CHECK_LOGIN_URLS = [
    "^/static",
    "^/favicon.ico",
]

# 分页的参数
PAGE_SIZE = 20 # 每页数据总数
PAGE_DISPLAY = 10 # 显示页码栏数量
```

```

13     # 解决中文在浏览器中显示的问题
14     JSON_AS_ASCII = False
15
16     AUTH_COOKIE_NAME = "food"
17
18     # 忽略登录验证的 路由
19     IGNORE_URLS = [
20         "^/user/login", # 登录路由
21     ]
22
23     # 忽略登录验证的 资源加载路由
24     IGNORE_CHECK_LOGIN_URLS = [
25         "^/static",
26         "^/favicon.icon",
27     ]
28
29     # 分页的参数
30     PAGE_SIZE = 20 # 每页数据总数
31     PAGE_DISPLAY = 10 # 显示页码栏数量
32
33
34
35

```

4. 用户列表视图实现

```

# -*- 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

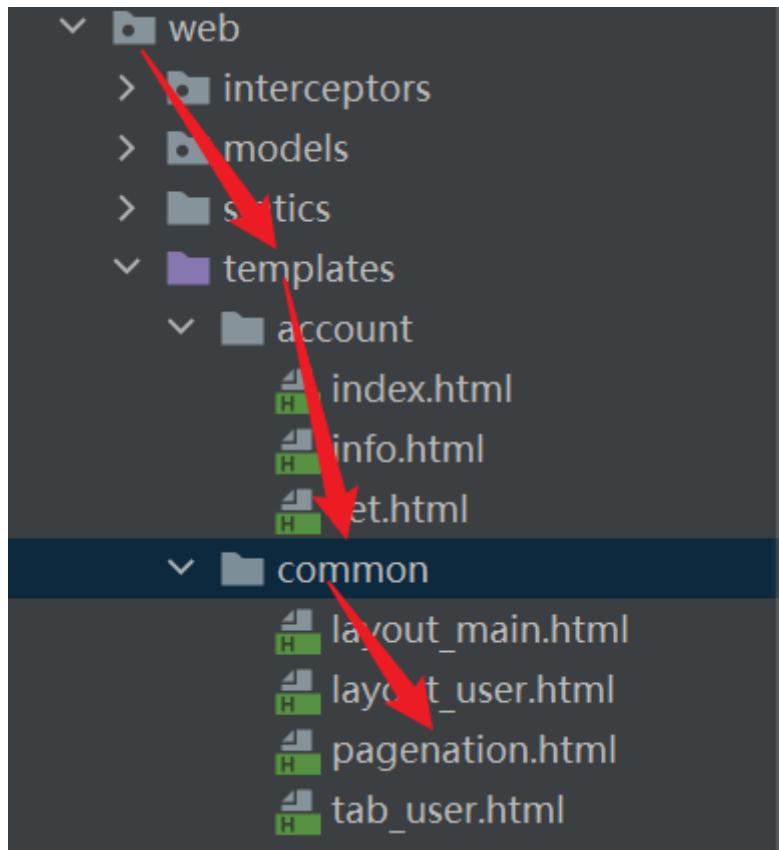
# 9. 构建响应数据
context = {
    "list": user_all[offset:limit], # 展示数据信息列表
    "pages": pages
}
# 10. 渲染响应
return render_template('account/index.html', **context)

route_account.add_url_rule('/index', view_func=Index.as_view('index'))

```

模板调整1

在web/templates/common目录内创建 pagination.html 文件



```
<div class="row">
    <div class="col-lg-12">
        <span class="pagination_count"
            style="line-height: 40px;">共{{ pages.total }}条记录 | 每页{{ pages.page_size }}条
        </span>
        <ul class="pagination pagination-lg pull-right" style="margin: 0 0 ;">
            {% if pages.is_prev == 1 %}
                <li>
                    <a href="{{ pages.url }}&p=1"><span>首页</span></a>
                </li>
            {% endif %}

            {% for idx in pages.range %}
                {% if idx == pages.current %}
                    <li class="active"><a href="javascript:void(0);">{{ idx }}</a></li>
                {% else %}
                    <li><a href="{{ pages.url }}&p={{ idx }}"/>{{ idx }}</a></li>
                {% endif %}
            {% endfor %}
            {% if pages.is_next == 1 %}
                <li>
                    <a href="{{ pages.url }}&p={{ pages.total_pages }}"/><span>尾页</span></a>
                </li>
            {% endif %}
        </ul>
    </div>
</div>
```

模板调整2

The screenshot shows the PyCharm IDE interface. On the left is the Project Explorer, displaying the file structure of the 'flpro' project. A red arrow points from the 'index.html' file in the 'account' directory to the code editor. The code editor contains the 'index.html' template, which includes a table with columns for operation, uid, nickname, mobile, email, and edit/delete links. The code uses Jinja2 syntax with {% if %} and {% for %} loops.

```
<th>操作</th>
</tr>
</thead>
<tbody>
    {% 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') }}">
                        <i class="fa fa-edit fa-lg"></i>
                    </a>
                    <a class="m-l remove" href="javascript:void(0);">
                        <i class="fa fa-trash fa-lg"></i>
                    </a>
                </td>
            </tr>
        {% endfor %}
    {% endif %}
</tbody>
```

```
{% extends "common/layout_main.html" %}
{% block content %}


<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 class="row">
    <div class="col-lg-12">
        <form class="form-inline wrap_search">
            <div class="row m-t p-w-m">
                <div class="form-group">
                    <select name="status" class="form-control inline">
                        <option value="-1">请选择状态</option>
                        <option value="1">正常</option>
                        <option value="0">已删除</option>
                    </select>
                </div>
                <div class="form-group">
                    <div class="input-group">
                        <input type="text" name="mix_kw" placeholder="请输入姓名或者手机号码" class="form-control" value="">
                        <input type="hidden" name="p" value="1">
                        <span class="input-group-btn">
                            <button type="button" class="btn btn-primary search">
                                <i class="fa fa-search"></i>搜索
                            </button>
                        </span>
                    </div>
                </div>
            </div>
        </form>
    </div>
</div>


```

```

                </button>
            </span>
        </div>
    </div>
    <hr>
    <div class="row">
        <div class="col-lg-12">
            <a class="btn btn-w-m btn-outline btn-primary pull-right"
               href="{{ buildUrl('/account/set') }}"
               ><i class="fa fa-plus"></i>账号
            </a>
        </div>
    </div>
</form>
<table class="table table-bordered m-t">
    <thead>
        <tr>
            <th>序号</th>
            <th>姓名</th>
            <th>手机</th>
            <th>邮箱</th>
            <th>操作</th>
        </tr>
    </thead>
    <tbody>
        {% 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') }}"
                           ><i class="fa fa-edit fa-lg"></i>
                        </a>

                        <a class="m-l remove" href="javascript:void(0);"
                           ><i class="fa fa-trash fa-lg"></i>
                        </a>
                    </td>
                </tr>
            {% endfor %}

            {% else %}
                <td colspan="5"> 暂无数据~</td>
            {% endif %}
        </tbody>
    </table>

    <!--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>#}
    {&gt; include "common/pagenation.html" %}
</div>
</div>
{&gt; endblock %}
```

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

The screenshot shows a web-based account management system. On the left is a dark sidebar with navigation links: 仪表盘 (Dashboard), 账号管理 (Account Management) (selected), 美餐管理 (Meal Management), 会员列表 (Member List), 财务管理 (Financial Management), and 统计管理 (Statistics Management). The main content area has a header with the text '欢迎使用编程浪子订餐管理后台' and a user icon. It features a search bar with dropdowns for '请选择状态' and '请输入姓名或者手机号码' and a '搜索' button. Below the search bar 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 with ID 1, Name 风助, Phone 11012345679, Email 111222@163.com, and operations icons for view, edit, and delete. The bottom of the page shows pagination information: '共1条记录 | 每页20条' and a page number '1'.

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

5. 展示用户个人信息

```
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
```

```
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()
```

```
# -*- 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)

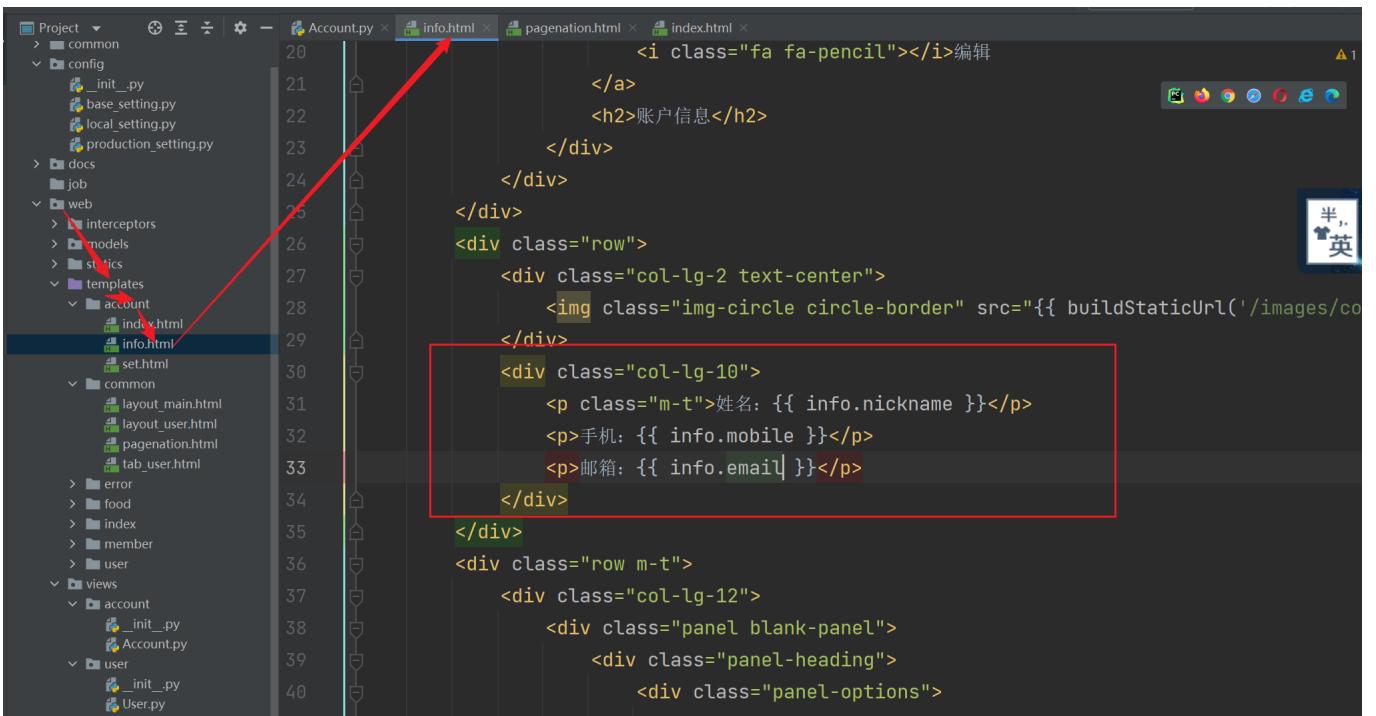
# 8. 有用户数据 就响应用户详情页
context["info"] = info

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

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

```

==调整模板==



The screenshot shows the PyCharm IDE interface. On the left is the project tree, displaying a directory structure with files like `Account.py`, `info.html` (which is selected), `pagination.html`, and `index.html`. A red arrow points from the file in the tree to the code editor on the right. The code editor shows the HTML template for displaying user information. A red box highlights the section of the template where user details (name, mobile, email) are listed. The status bar at the bottom indicates: "启动 访问 http://127.0.0.1:8888/account/index".

```

></i>编辑
<a href="#">></i>编辑
<h2>账户信息</h2>
</div>
</div>
<div class="row">
    <div class="col-lg-2 text-center">
        
        <p class="m-t">姓名: {{ info.nickname }}</p>
        <p>手机: {{ info.mobile }}</p>
        <p>邮箱: {{ info.email }}</p>
    </div>
<div class="row m-t">
    <div class="col-lg-12">
        <div class="panel blank-panel">
            <div class="panel-heading">
                <div class="panel-options">

```

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

账户列表

请选择状态 ▾ 输入姓名或者手机号 点击搜索

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

账户信息

姓名: 风勋
手机: 11012345679
邮箱: 111222@163.com

访问记录

访问时间	访问Url
暂无数据	