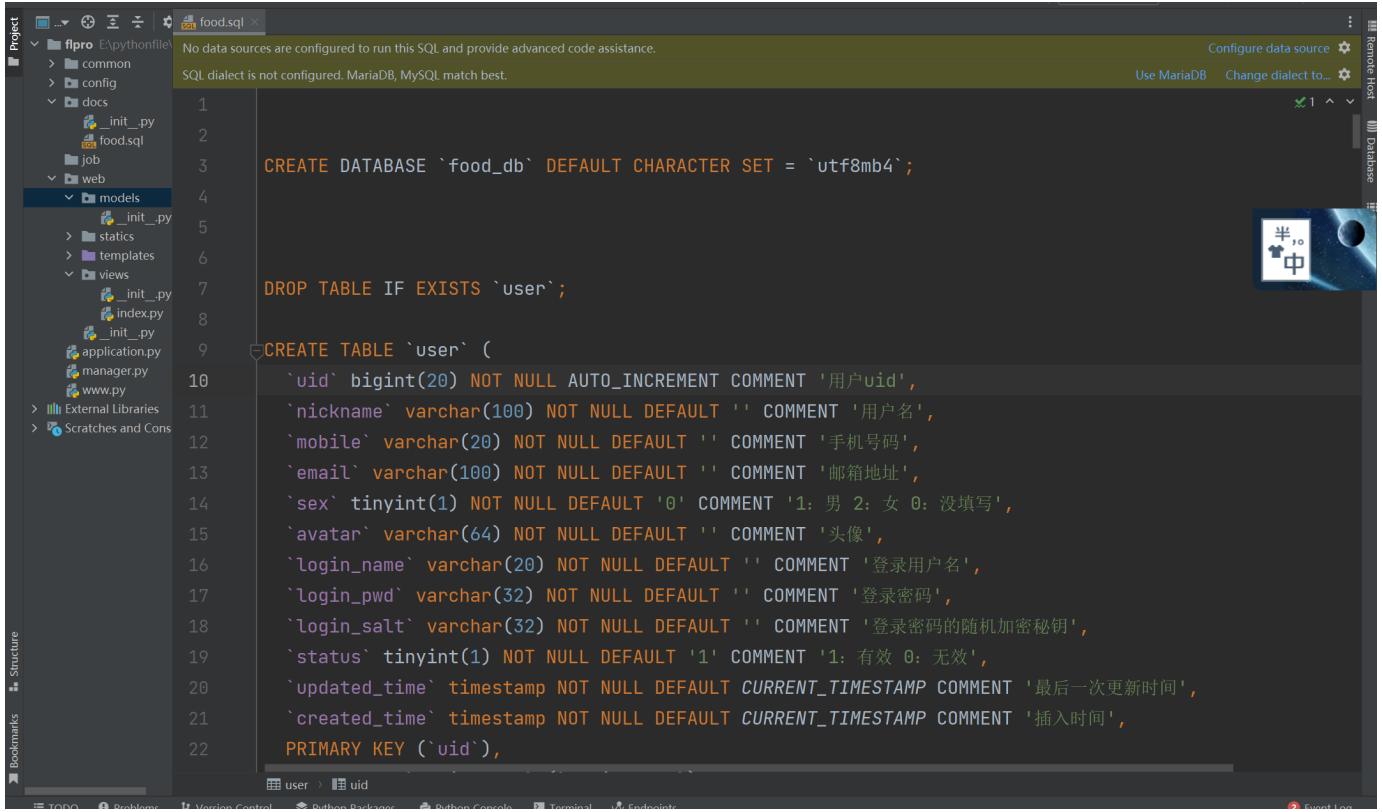


1.建立数据库表

这里我们根据测试数据库提供的food.sql包
使用pycharm打开



```
CREATE DATABASE `food_db` DEFAULT CHARACTER SET = `utf8mb4`;

DROP TABLE IF EXISTS `user`;

CREATE TABLE `user` (
    `uid` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '用户uid',
    `nickname` varchar(100) NOT NULL DEFAULT '' COMMENT '用户名',
    `mobile` varchar(20) NOT NULL DEFAULT '' COMMENT '手机号码',
    `email` varchar(100) NOT NULL DEFAULT '' COMMENT '邮箱地址',
    `sex` tinyint(1) NOT NULL DEFAULT '0' COMMENT '1: 男 2: 女 0: 没填写',
    `avatar` varchar(64) NOT NULL DEFAULT '' COMMENT '头像',
    `login_name` varchar(20) NOT NULL DEFAULT '' COMMENT '登录用户名',
    `login_pwd` varchar(32) NOT NULL DEFAULT '' COMMENT '登录密码',
    `login_salt` varchar(32) NOT NULL DEFAULT '' COMMENT '登录密码的随机加密秘钥',
    `status` tinyint(1) NOT NULL DEFAULT '1' COMMENT '1: 有效 0: 无效',
    `updated_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '最后一次更新时间',
    `created_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '插入时间',
    PRIMARY KEY (`uid`),
    UNIQUE KEY `login_name` (`login_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='用户表（管理员）';
```

打开后复制如下部分

```
CREATE TABLE `user` (
    `uid` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '用户uid',
    `nickname` varchar(100) NOT NULL DEFAULT '' COMMENT '用户名',
    `mobile` varchar(20) NOT NULL DEFAULT '' COMMENT '手机号码',
    `email` varchar(100) NOT NULL DEFAULT '' COMMENT '邮箱地址',
    `sex` tinyint(1) NOT NULL DEFAULT '0' COMMENT '1: 男 2: 女 0: 没填写',
    `avatar` varchar(64) NOT NULL DEFAULT '' COMMENT '头像',
    `login_name` varchar(20) NOT NULL DEFAULT '' COMMENT '登录用户名',
    `login_pwd` varchar(32) NOT NULL DEFAULT '' COMMENT '登录密码',
    `login_salt` varchar(32) NOT NULL DEFAULT '' COMMENT '登录密码的随机加密秘钥',
    `status` tinyint(1) NOT NULL DEFAULT '1' COMMENT '1: 有效 0: 无效',
    `updated_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '最后一次更新时间',
    `created_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '插入时间',
    PRIMARY KEY (`uid`),
    UNIQUE KEY `login_name` (`login_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='用户表（管理员）';
```

进入数据库，建表

```
Database
+-----+
information_schema
anli
anli2
db_books
demo_book
fldemo
food_db
mysql
performance_schema
py
test
v2ex
xiaodemo
xiaoyu_demo
+-----+
14 rows in set (0.02 sec)

mysql> use food_db
Database changed
mysql> CREATE TABLE user (
    `uid` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '用户uid',
    `nickname` varchar(100) NOT
NULL DEFAULT '' COMMENT '用户名',
    `mobile` varchar(20) NOT NULL DEFAULT '' COMMENT '手机号码',
    `email` varchar(100)
NOT NULL DEFAULT '' COMMENT '邮箱地址',
    `sex` tinyint(1) NOT NULL DEFAULT '0' COMMENT '1: 男 2: 女 0: 没填写',
    `avatar` varchar(64) NOT NULL DEFAULT '' COMMENT '头像',
    `login_name` varchar(20) NOT NULL DEFAULT '' COMMENT '登录用户名',
    `login_pwd` varchar(32) NOT NULL DEFAULT '' COMMENT '登录密码',
    `login_salt` varchar(32) NOT NULL DEFAULT '' COMM
ENT '登录密码的随机加密秘钥',
    `status` tinyint(1) NOT NULL DEFAULT '1' COMMENT '1: 有效 0: 无效',
    `updated_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '最后一次更新时间',
    `created_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '插入时间',
    PRIMARY KEY (`uid`),
    UNIQUE KEY `login_name`(`login_name`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COMMENT='用户表(管理员)';
Query OK, 0 rows affected (0.03 sec)

mysql>
```



查看

```
mysql> show tables;
+-----+
| Tables_in_food_db |
+-----+
| user              |
+-----+
1 row in set (0.01 sec)
```

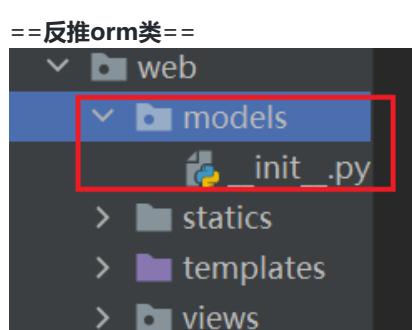
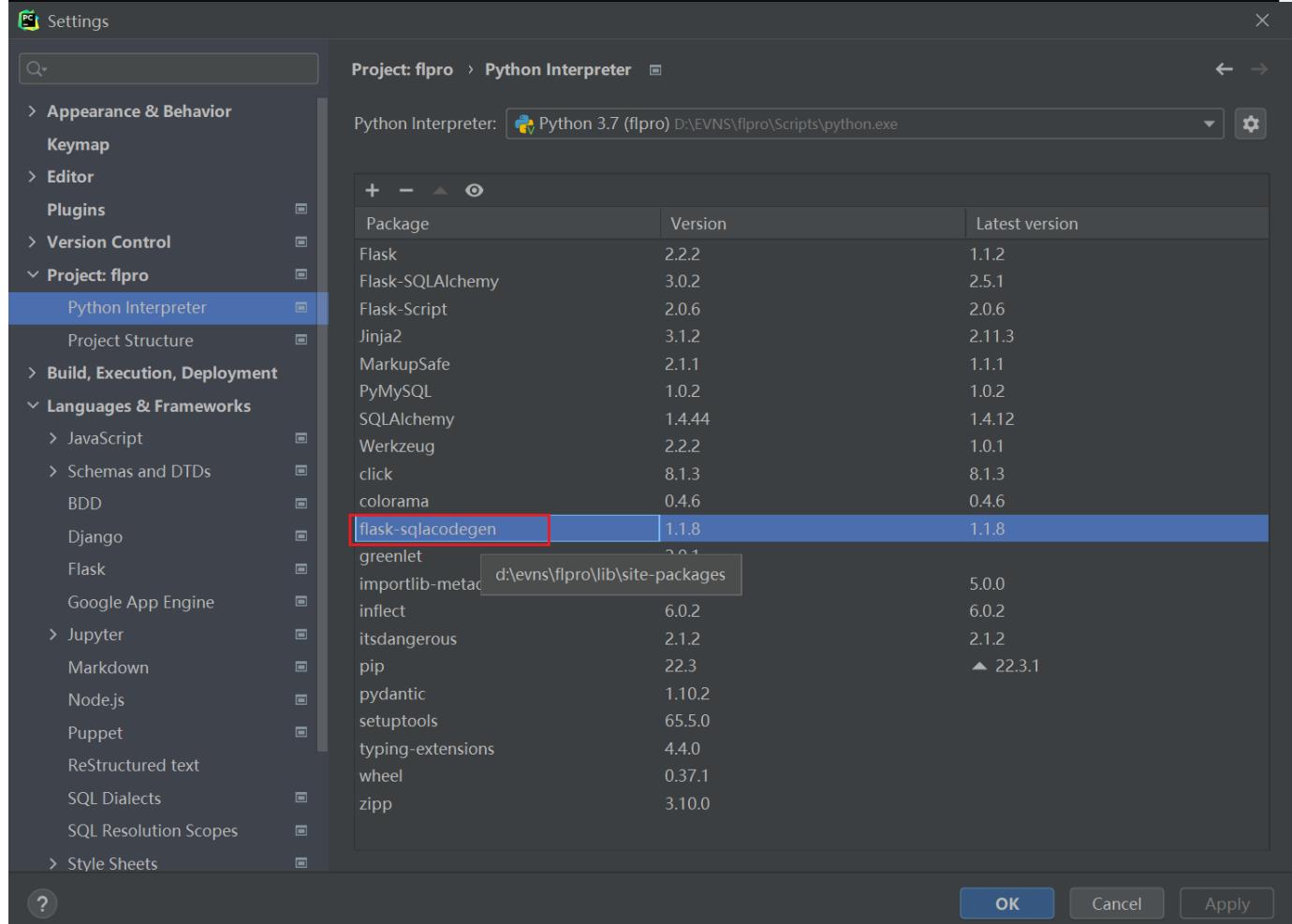
2. 自动反推ORM类

==安装==

```
flask-sqlacodegen --- 此模块根据表自动反推ORM类
pip install flask-sqlacodegen -i https://pypi.tuna.tsinghua.edu.cn/simple
```

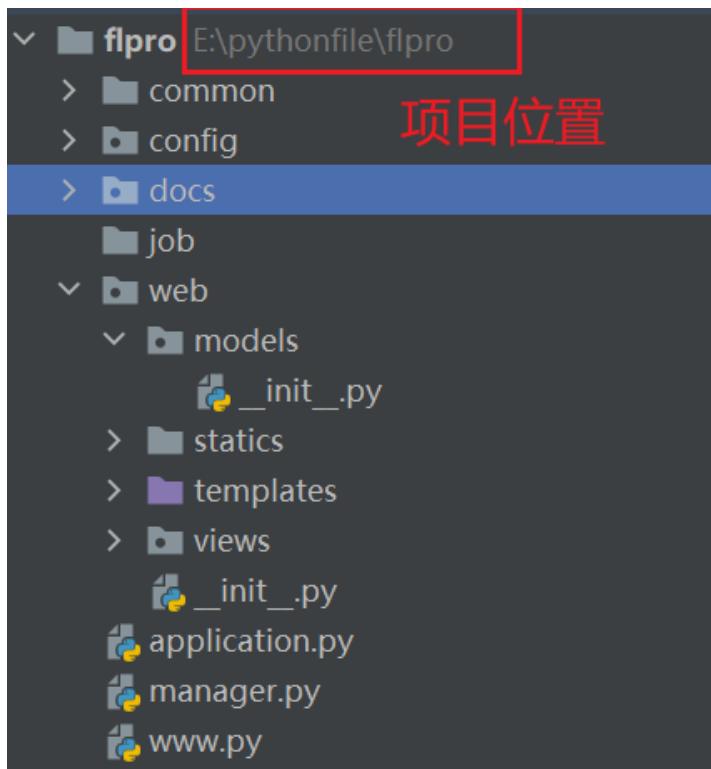
```
(flpro) E:\pythonfile\pro_food>pip install flask-sqlacodegen -i https://pypi.tuna.tsinghua.edu.cn/simple
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting flask-sqlacodegen
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/1d/c6/8291762360b426ed46b6d90765a3037de48862320586e50d21c71bb
aa51d/flask_sqlacodegen-1.1.8-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: SQLAlchemy>=0.6.0 in d:\evns\flpro\lib\site-packages (from flask-sqlacodegen)
Collecting inflect>=0.2.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/67/e2/bcd7099b31d6a1f7be358f7ef7cf6fc97cc5a66353784fd400e1
243fb/inflect-6.0.2-py3-none-any.whl (34 kB)
Collecting pydantic>=1.9.1
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/dc/bf/5965230bf0547c5fa0005984564146dcc414e6e8b6349177eca4137
61013/pydantic-1.10.2-cp37-cp37m-win_amd64.whl (2.1 MB)
    2.1/2.1 MB 1.4 MB/s eta 0:00:00
Requirement already satisfied: importlib-metadata in d:\evns\flpro\lib\site-packages (from SQLAlchemy>=0.6.0>flask-sqlacodegen) (5.0.0)
Requirement already satisfied: greenlet!=0.4.17 in d:\evns\flpro\lib\site-packages (from SQLAlchemy>=0.6.0>flask-sqlacodegen) (2.0.1)
Requirement already satisfied: typing-extensions>=4.1.0 in d:\evns\flpro\lib\site-packages (from pydantic>=1.9.1>inflect>=0.2.0>flask-sqlacodegen) (4.4.0)
Requirement already satisfied: zipp>=0.5 in d:\evns\flpro\lib\site-packages (from importlib-metadata->SQLAlchemy>=0.6.0>flask-sqlacodegen) (3.10.0)
Installing collected packages: pydantic, inflect, flask-sqlacodegen
Successfully installed flask-sqlacodegen-1.1.8 inflect-6.0.2 pydantic-1.10.2
```

[notice] A new release of pip available: 22.3 → 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip



```
flask-sqlacodegen mysql+pymysql://root:qwe123@127.0.0.1/food_db --tables user --outfile "web/models/User.py" --flask  
# 因为使用pymysql连接，因此mysql+pymysql  
# flask-sqlacodegen mysql+pymysql://mysql用户名:用户密码@127.0.0.1/food_db --tables 目标表名 --outfile "指定存放文件位置" --flask
```

找到项目位置

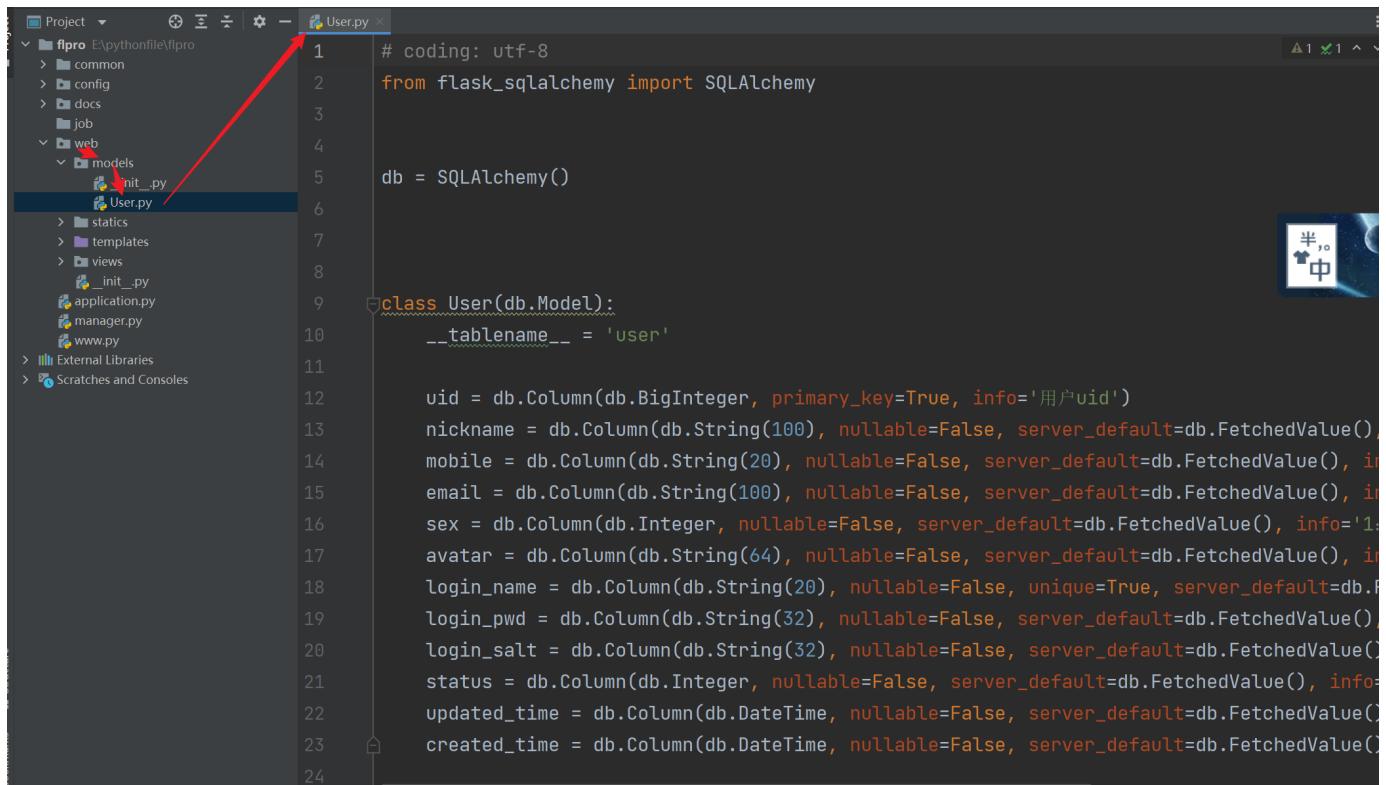


找到相应位置，执行推导命令

```
# 使用虚拟环境，注意虚拟环境切换
```

```
(flpro) E:\pythonfile\flpro>flask-sqlacodegen mysql+pymysql://root:qwe123@127.0.0.1/food_db --tables user --outfile "web/models/User.py" --flask  
(flpro) E:\pythonfile\flpro>
```

执行完毕无异常后 回到项目中查看



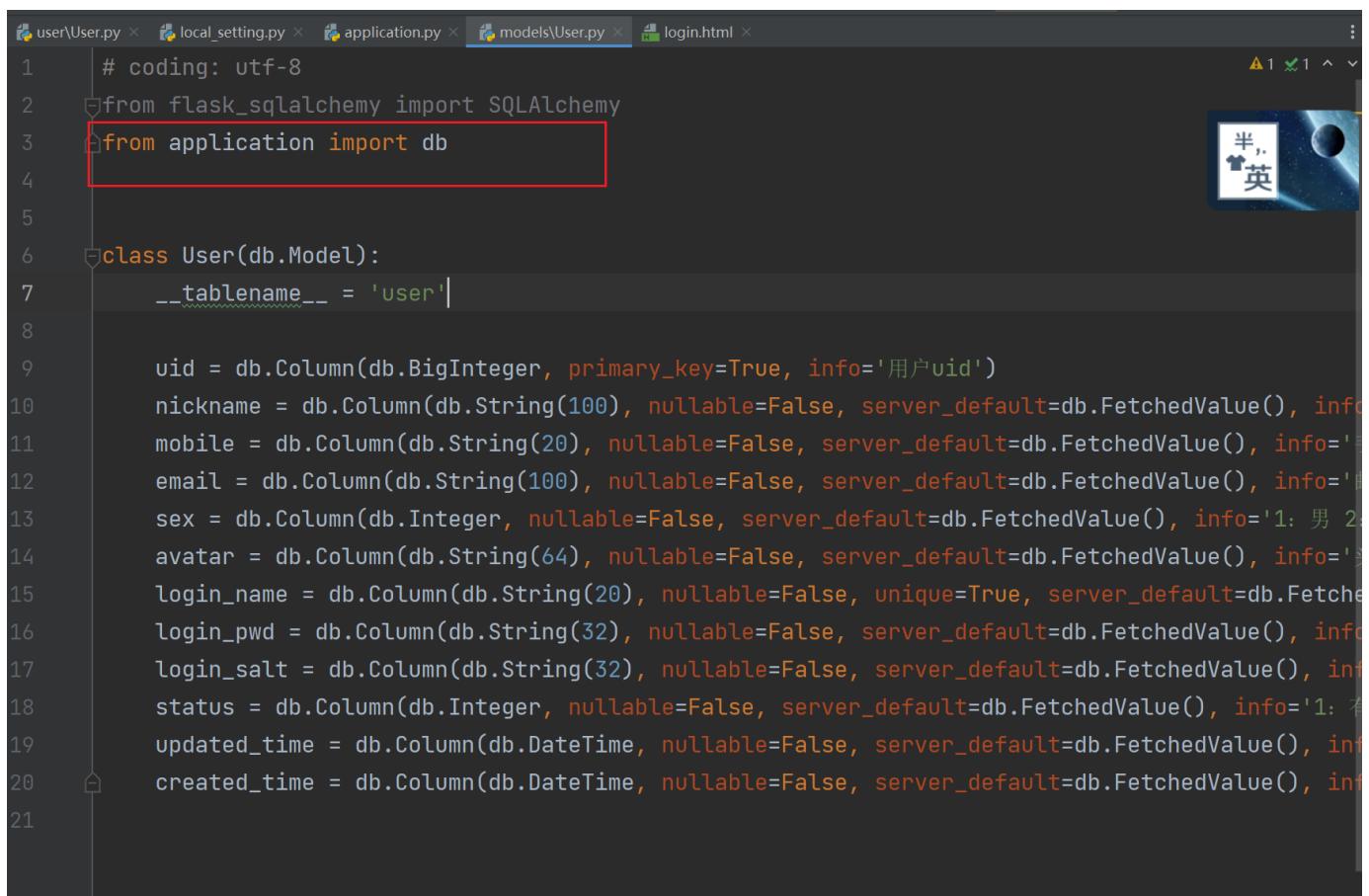
```
# coding: utf-8
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class User(db.Model):
    __tablename__ = 'user'

    uid = db.Column(db.BigInteger, primary_key=True, info='用户uid')
    nickname = db.Column(db.String(100), nullable=False, server_default=db.FetchedValue(), info='用户名')
    mobile = db.Column(db.String(20), nullable=False, server_default=db.FetchedValue(), info='手机号')
    email = db.Column(db.String(100), nullable=False, server_default=db.FetchedValue(), info='邮箱')
    sex = db.Column(db.Integer, nullable=False, server_default=db.FetchedValue(), info='1: 男 2: 女')
    avatar = db.Column(db.String(64), nullable=False, server_default=db.FetchedValue(), info='头像')
    login_name = db.Column(db.String(20), nullable=False, unique=True, server_default=db.FetchedValue())
    login_pwd = db.Column(db.String(32), nullable=False, server_default=db.FetchedValue())
    login_salt = db.Column(db.String(32), nullable=False, server_default=db.FetchedValue())
    status = db.Column(db.Integer, nullable=False, server_default=db.FetchedValue(), info='状态')
    updated_time = db.Column(db.DateTime, nullable=False, server_default=db.FetchedValue())
    created_time = db.Column(db.DateTime, nullable=False, server_default=db.FetchedValue())
```

此时`db`对象时独立实例的，这与项目毫无关系，因此我们需要使用我们项目中的`db`实例对象



```
# coding: utf-8
from flask_sqlalchemy import SQLAlchemy
from application import db

class User(db.Model):
    __tablename__ = 'user'

    uid = db.Column(db.BigInteger, primary_key=True, info='用户uid')
    nickname = db.Column(db.String(100), nullable=False, server_default=db.FetchedValue(), info='用户名')
    mobile = db.Column(db.String(20), nullable=False, server_default=db.FetchedValue(), info='手机号')
    email = db.Column(db.String(100), nullable=False, server_default=db.FetchedValue(), info='邮箱')
    sex = db.Column(db.Integer, nullable=False, server_default=db.FetchedValue(), info='1: 男 2: 女')
    avatar = db.Column(db.String(64), nullable=False, server_default=db.FetchedValue(), info='头像')
    login_name = db.Column(db.String(20), nullable=False, unique=True, server_default=db.FetchedValue())
    login_pwd = db.Column(db.String(32), nullable=False, server_default=db.FetchedValue())
    login_salt = db.Column(db.String(32), nullable=False, server_default=db.FetchedValue())
    status = db.Column(db.Integer, nullable=False, server_default=db.FetchedValue(), info='状态')
    updated_time = db.Column(db.DateTime, nullable=False, server_default=db.FetchedValue())
    created_time = db.Column(db.DateTime, nullable=False, server_default=db.FetchedValue())
```

3. 用户数据导入

```
INSERT INTO `user` (`uid`, `nickname`, `mobile`, `email`, `sex`, `avatar`, `login_name`, `login_pwd`, `login_salt`, `status`, `updated_time`, `created_time`)
VALUES (1, '风勋帅')
```

```
帅', '11012345679', '111111@163.com', 1, '', 'fengxun', '816440c40b7a9d55ff9eb7b20760862c', 'cF3JfH5FJfQ8B2Ba', 1, '2021-03-15 14:08:48', '2021-03-15 14:08:48');
```

```
mysql> use food_db
Database changed
mysql> INSERT INTO `user`(`uid`, `nickname`, `mobile`, `email`, `sex`, `avatar`, `login_name`, `login_pwd`, `login_salt`, `status`, `updated_time`, `created_time`) VALUES (1, '风勋帅帅', '11012345679', '111111@163.com', 1, 'fengxun', '816440c40b7a9d55ff9eb7b20760862c', 'cF3JfH5FJfQ8B2Ba', 1, '2021-03-15 14:08:48', '2021-03-15 14:08:48');
Query OK, 1 row affected (0.00 sec)

mysql>
```

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> use food_db
Database changed
mysql> INSERT INTO `user`(`uid`, `nickname`, `mobile`, `email`, `sex`, `avatar`, `login_name`, `login_pwd`, `login_salt`, `status`, `updated_time`, `created_time`) VALUES (1, '风勋帅帅', '11012345679', '111111@163.com', 1, 'fengxun', '816440c40b7a9d55ff9eb7b20760862c', 'cF3JfH5FJfQ8B2Ba', 1, '2021-03-15 14:08:48', '2021-03-15 14:08:48');
Query OK, 1 row affected (0.00 sec)

mysql> select * from user;
+-----+-----+-----+-----+-----+-----+-----+-----+
| uid | nickname | mobile | email | sex | avatar | login_name | login_pwd |
| login_salt | status | updated_time | created_time |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 风勋帅帅 | 11012345679 | 111111@163.com | 1 | fengxun | 816440c40b7a9d55ff9eb7b20760862c | cF3JfH5FJfQ8B2Ba |
| 1 | 2021-03-15 14:08:48 | 2021-03-15 14:08:48 |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

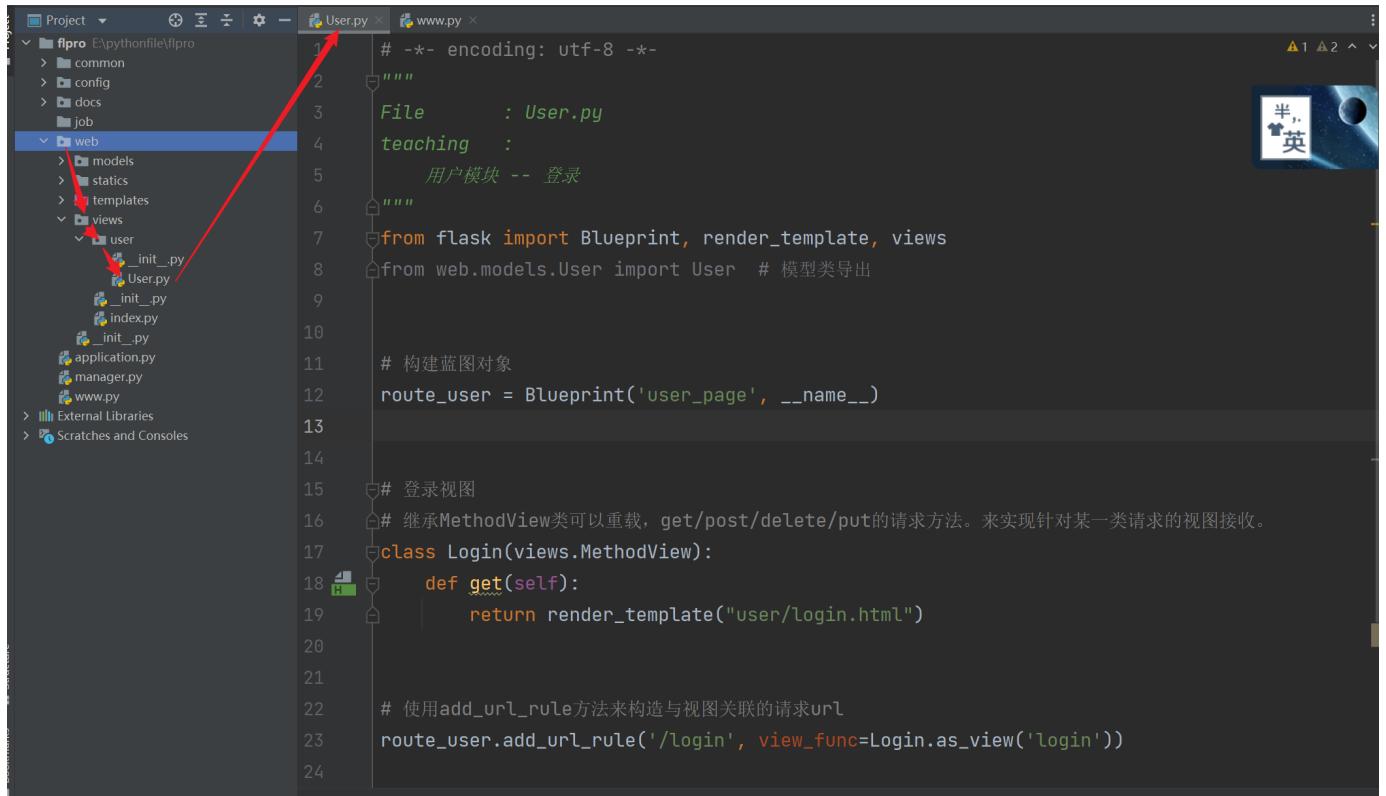
用户名为 fengxun
密码为 123456

4. 登录视图建立

函数视图在目前开发中便捷的确好，但是其维护性不太好

因此 类视图 优先选择

那么同学们注意类视图结合蓝图的操作



The screenshot shows the PyCharm IDE interface. On the left, the project structure is displayed with a tree view of files and folders. A red arrow points from the 'User.py' entry in the tree to the 'User.py' tab in the top navigation bar. The main editor area contains the code for 'User.py'. The code is as follows:

```
# -*- encoding: utf-8 -*-
"""
File      : User.py
teaching   :
    用户模块 -- 登录
"""

from flask import Blueprint, render_template, views
from web.models.User import User # 模型类导出

# 构建蓝图对象
route_user = Blueprint('user_page', __name__)

# 登录视图
# 继承MethodView类可以重载, get/post/delete/put的请求方法。来实现针对某一类请求的视图接收。
class Login(views.MethodView):
    def get(self):
        return render_template("user/login.html")

# 使用add_url_rule方法来构造与视图关联的请求url
route_user.add_url_rule('/login', view_func=Login.as_view('login'))
```

```
# -*- encoding: utf-8 -*-
"""
File      : User.py
teaching   :
    用户模块 -- 登录
"""

from flask import Blueprint, render_template, views
from web.models.User import User # 模型类导出

# 构建蓝图对象
route_user = Blueprint('user_page', __name__)

# 登录视图
# 继承MethodView类可以重载, get/post/delete/put的请求方法。来实现针对某一类请求的视图接收。
class Login(views.MethodView):
    def get(self):
        return render_template("user/login.html")

# 使用add_url_rule方法来构造与视图关联的请求url
# 继承自模板的as_view方法可以将类转换为可以为路由注册的视图函数
# as_view('login') --- 传入的是站点名 (标准操作是视图名小写 Login -- login)
#     <Rule '/user/login' (GET, HEAD, OPTIONS) -> user_page.login>] user_page.login中的login就是as_view指明的站点名
route_user.add_url_rule('/login', view_func=Login.as_view('login'))
```

==注册蓝图对象==

```
# -*- 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

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

```
# -*- 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

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

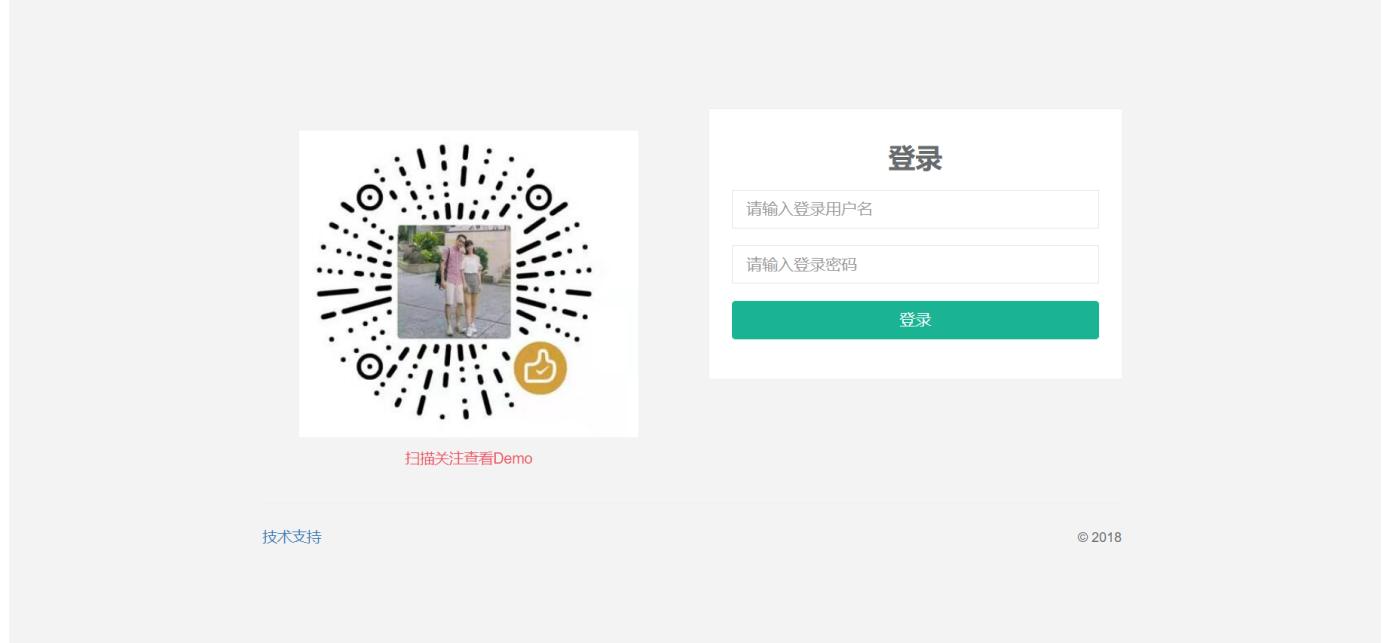
启动

The screenshot shows the PyCharm IDE interface. The project structure on the left includes a 'fipro' project with subfolders like 'common', 'config', 'docs', 'job', 'web', 'models', 'statics', 'templates', and 'views'. Under 'views', there's a 'user' folder containing files like '_init_.py', 'User.py', '_init_.py', 'index.py', '_init_.py', and 'application.py'. The 'www.py' file in the main editor window contains code for registering blueprints:

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

The 'Run' tab at the bottom shows the server starting with the message: "Serving Flask app 'application'". It also displays a warning about using a development server in production and provides the URL <http://127.0.0.1:8888>.

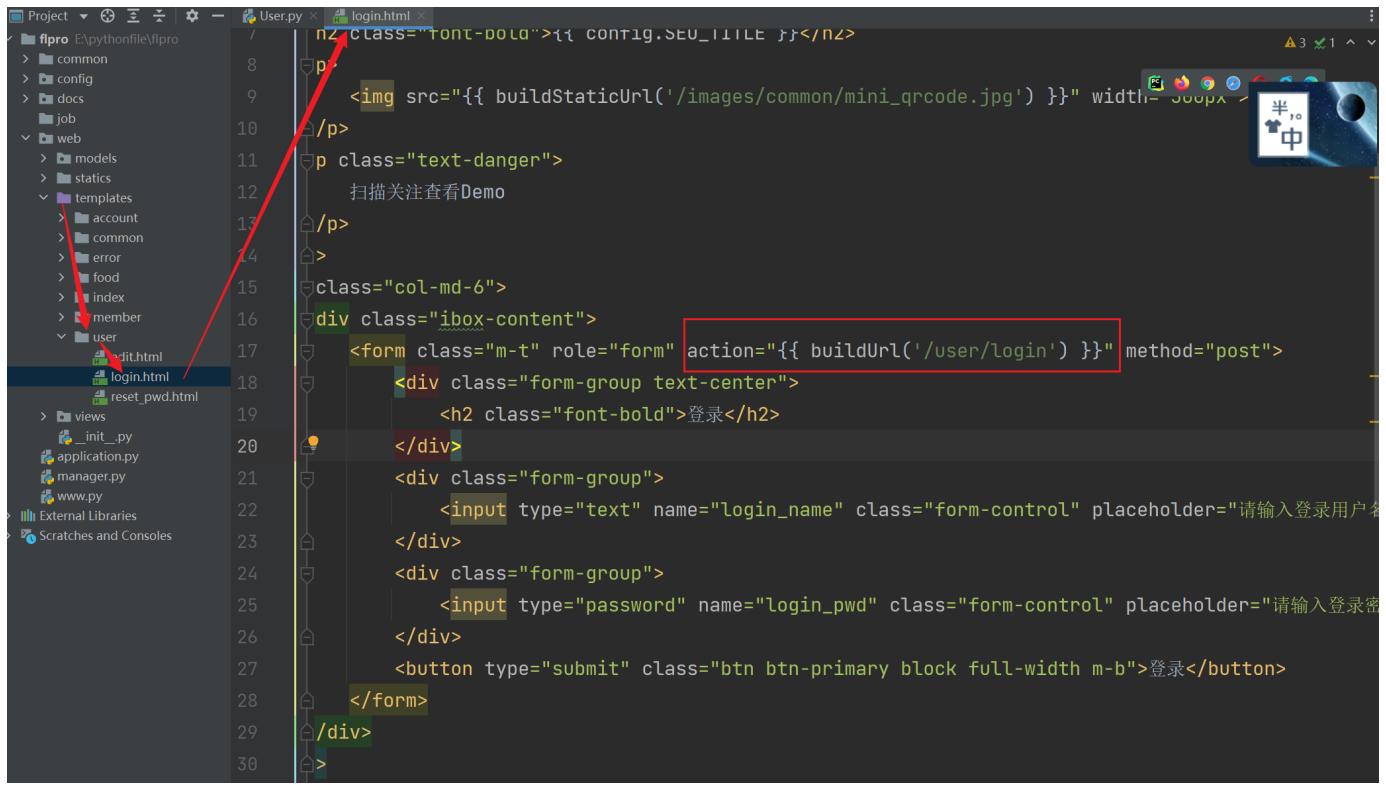
访问 <http://127.0.0.1:8888/user/login>



5. 登录表单分析

==分析提交方式==

根据login.html分析我们可以知道 --- 此处采用的表单数据提交方式为默认方式并未通过ajax进行处理



```
Project -> flpro E:\pythonfile\flpro
  > common
  > config
  > docs
  > job
  > web
    > models
    > statics
    > templates
      > account
      > common
      > error
      > food
      > index
      > member
      > user
        > edit.html
        > login.html
        > reset_pwd.html
    > views
      > __init__.py
      > application.py
      > manager.py
      > www.py
  > External Libraries
  > Scratches and Consoles
```

User.py

```
7     nz><class="font-bold">{{ config.SECURITY_TITLE }}</n>
8   > p>
9     
10    > p class="text-danger">
11      扫描关注查看Demo
12    > /p>
13    > >
14    > class="col-md-6">
15      > div class="ibox-content">
16        <form class="m-t" role="form" action="{{ buildUrl('/user/login') }}" method="post">
17          <div class="form-group text-center">
18            <h2 class="font-bold">登录</h2>
19          </div>
20          <div class="form-group">
21            <input type="text" name="login_name" class="form-control" placeholder="请输入登录用户名" />
22          </div>
23          <div class="form-group">
24            <input type="password" name="login_pwd" class="form-control" placeholder="请输入登录密码" />
25          </div>
26          <button type="submit" class="btn btn-primary block full-width m-b">登录</button>
27        </form>
28      > /div>
29    > /div>
30  > /div>
```

这里有个buildUrl方法：这是我们注册的全局模板函数 --- 参考《1.项目搭建布局》第5点 全局模板函数

==分析数据键==

```
> div class="ibox-content">
  <form class="m-t" role="form" action="{{ buildUrl('/user/login') }}" method="post">
    <div class="form-group text-center">
      <h2 class="font-bold">登录</h2>
    </div>
    <div class="form-group">
      <input type="text" name="login_name" class="form-control" placeholder="请输入登录用户名" />
    </div>
    <div class="form-group">
      <input type="password" name="login_pwd" class="form-control" placeholder="请输入登录密码" />
    </div>
    <button type="submit" class="btn btn-primary block full-width m-b">登录</button>
  </form>
</div>
```

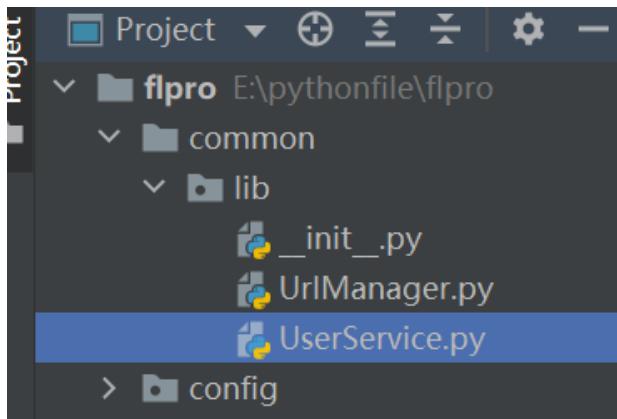
6.数据库密码加密

正式开始之前我们需要 还原用户密码 加密过程

因为密码数据加密是不可逆的

本次项目的数据由于是提前定好的，所以我们就直接还原过程

common/lib目录下建立UserService.py文件



```
# -*-coding:utf-8-*-
# common/lib/UserService.py
import hashlib, base64

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


if __name__ == '__main__':
    # 加密后数据: 816440c40b7a9d55ff9eb7b20760862c
    # 加密盐: cF3JfH5FJfQ8B2Ba
    # 原数据: 123456
    print(UserService.genPwd('123456', 'cF3JfH5FJfQ8B2Ba')) # 816440c40b7a9d55ff9eb7b20760862c
```

8.登录实现

```
13     route_user = Blueprint('user_page', __name__)
14
15     # 登录视图
16     # 继承MethodView类可以重载, get/post/delete/put的请求方法。来实现针对某一类请求的视图接收。
17     class Login(MethodView):
18         def get(self):
19             return render_template("user/login.html")
20
21         def post(self):
22             # 1.注意导入请求上下文 request from flask import request
23
24             # 2.获取form表单数据
25             data = request.values
26
27             # 3.获取用户名及密码
28             login_name = data.get('login_name')
29             login_pwd = data.get('login_pwd')
30
31             # 4.构建响应数据对象
32             resp = {
33                 "code": 200, # 状态码
34                 "msg": "登录成功", # 响应信息
35             }
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
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
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
```

```
# -*- encoding: utf-8 -*-
"""
File       : User.py
teaching   :
    用户模块 -- 登录
"""

from flask import Blueprint, render_template, views, request, jsonify, url_for, redirect
from web.models.User import User # 模型类导出
from common.lib.UserService import UserService

# 构建蓝图对象
route_user = Blueprint('user_page', __name__)

# 登录视图
# 继承MethodView类可以重载, get/post/delete/put的请求方法。来实现针对某一类请求的视图接收。
class Login(views.MethodView):
    def get(self):
        return render_template("user/login.html")

    def post(self):
        # 1.注意导入请求上下文 request from flask import request

        # 2.获取form表单数据
        data = request.values

        # 3.获取用户名及密码
        login_name = data.get('login_name')
        login_pwd = data.get('login_pwd')

        # 4.构建响应数据对象
        resp = {
            "code": 200, # 状态码
            "msg": "登录成功", # 响应信息
            "data": {} # 响应数据
        }

        # 5.判断数据是否获取及其是否满足要求
        if (not login_name) or len(login_name) < 1:
```

```

# 用户名没有数据 或者 用户名长度小于1
resp["code"] = -1
resp["msg"] = "请输入正确的登录用户名"
# 导入 jsonify --- from flask import jsonify
return jsonify(resp)

if (not login_pwd) or len(login_pwd) < 1:
    # 密码没有数据 或者 密码长度小于1
    resp["code"] = -1
    resp["msg"] = "请输入正确的密码"
    return jsonify(resp) # 响应json数据

# 6.根据用户名查询用户是否存在于数据库
user_info = User.query.filter_by(login_name=login_name).first()
if not user_info: # 数据库查不到 -- 即不存在
    resp["code"] = -1
    resp["msg"] = "请输入正确的用户名或密码"
    return jsonify(resp) # 响应json数据

# 7.用户存在 -- 加密密码
salt = user_info.login_salt # 根据user_info对象数据 获取数据库中的用户密码 加密盐
# 加密方法 from common.lib.UserService import UserService
gen_pwd = UserService.genPwd(login_pwd, salt) # 跟注册时相同的加密机制 -- 获得加密数据

# 8.将秘密加密数据 与 数据库密码数据进行比对

if user_info.login_pwd != gen_pwd:
    # 不相同, 说明密码或者用户名不对
    resp["code"] = -1
    resp["msg"] = "请输入正确的用户名或密码"
    return jsonify(resp)

# 9.登录成功 重定向首页 from flask import url_for, redirect
return redirect(url_for('index_page.index'))

# 使用add_url_rule方法来构造与视图关联的请求url
# 继承自模板的as_view方法可以将类转换为可以为路由注册的视图函数
# as_view('login') --- 传入的是站点名 (标准操作是视图名小写 Login -- login)
#       <Rule '/user/login' (GET, HEAD, OPTIONS) -> user_page.login>] user_page.login中的login就是as_view指明的站点名
route_user.add_url_rule('/login', view_func=Login.as_view('login'))

```

完成后记得-启动登录测试

数据库中的用户： fengxun 密码： 123456