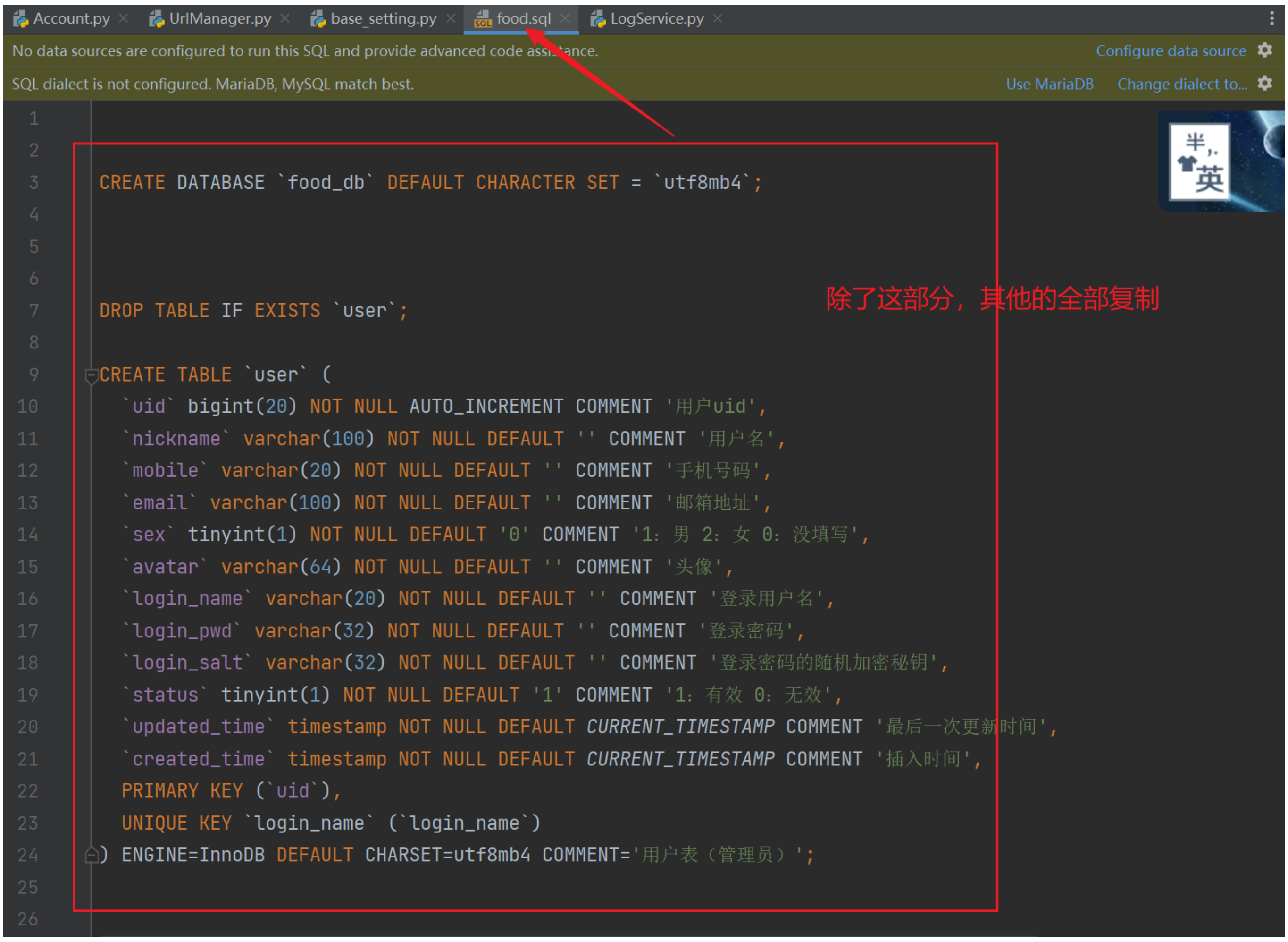
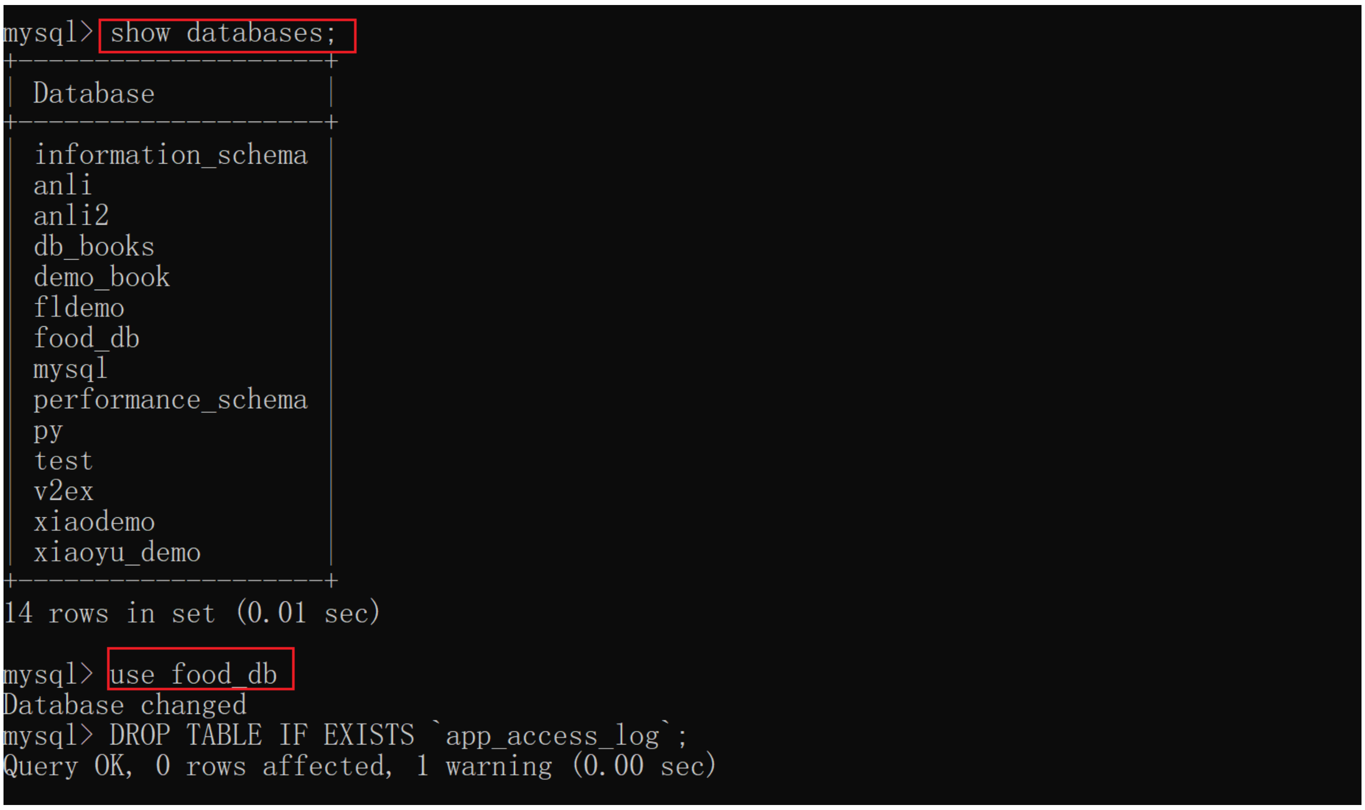
1.建立数据库 - 表

在初期的food.sql文件中复制以下数据 ---- 一定要从sql文件中去复制

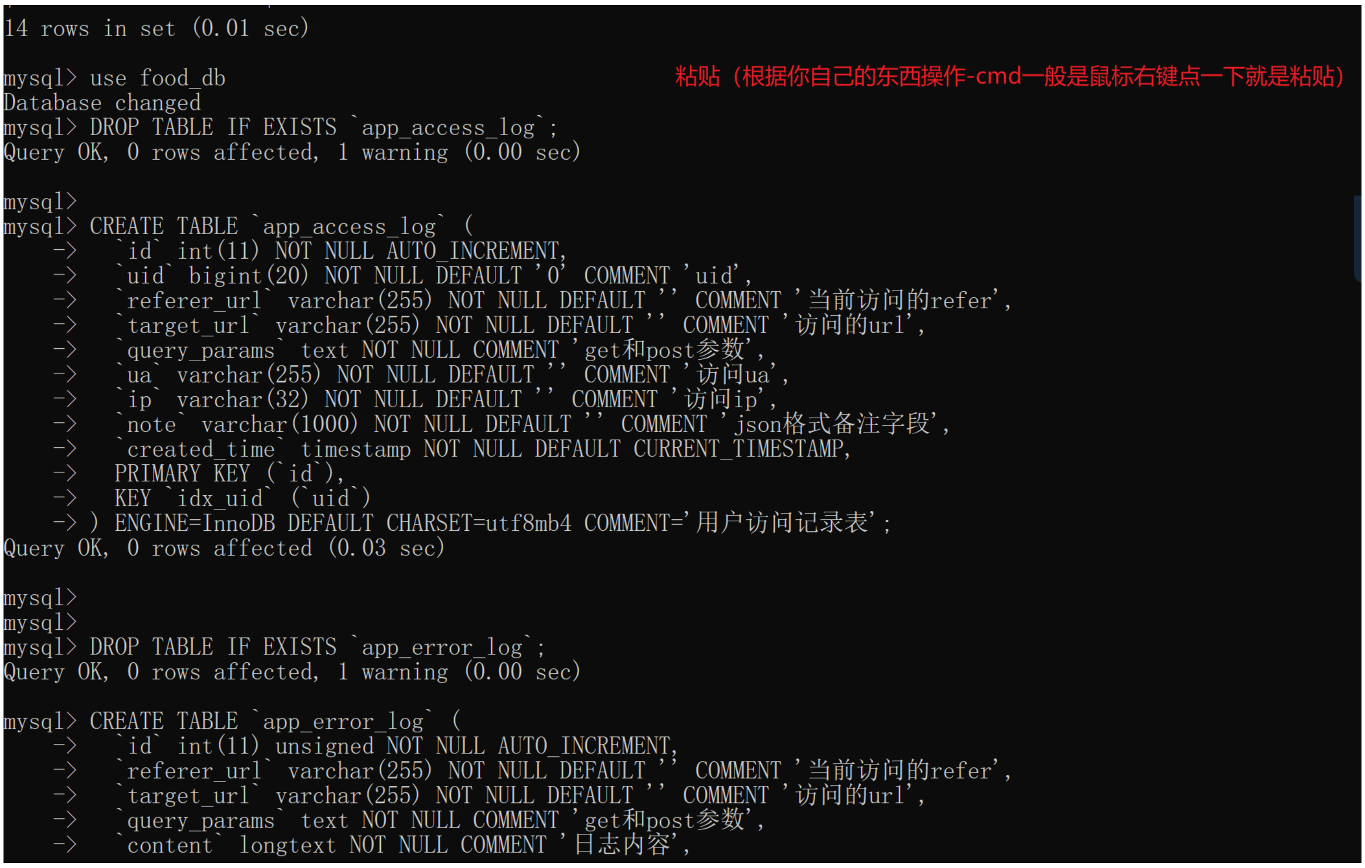

Account.py
UrlManager.py
base_setting.py
food.sql
LogService.py
No data sources are configured to run this SQL and provide advanced code assistance.
Configure data source
SQL dialect is not configured. MariaDB, MySQL match best.
Use MariaDB
Change dialect to...
除了这部分，其他的全部复制

```sql

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）';

```

进入数据库

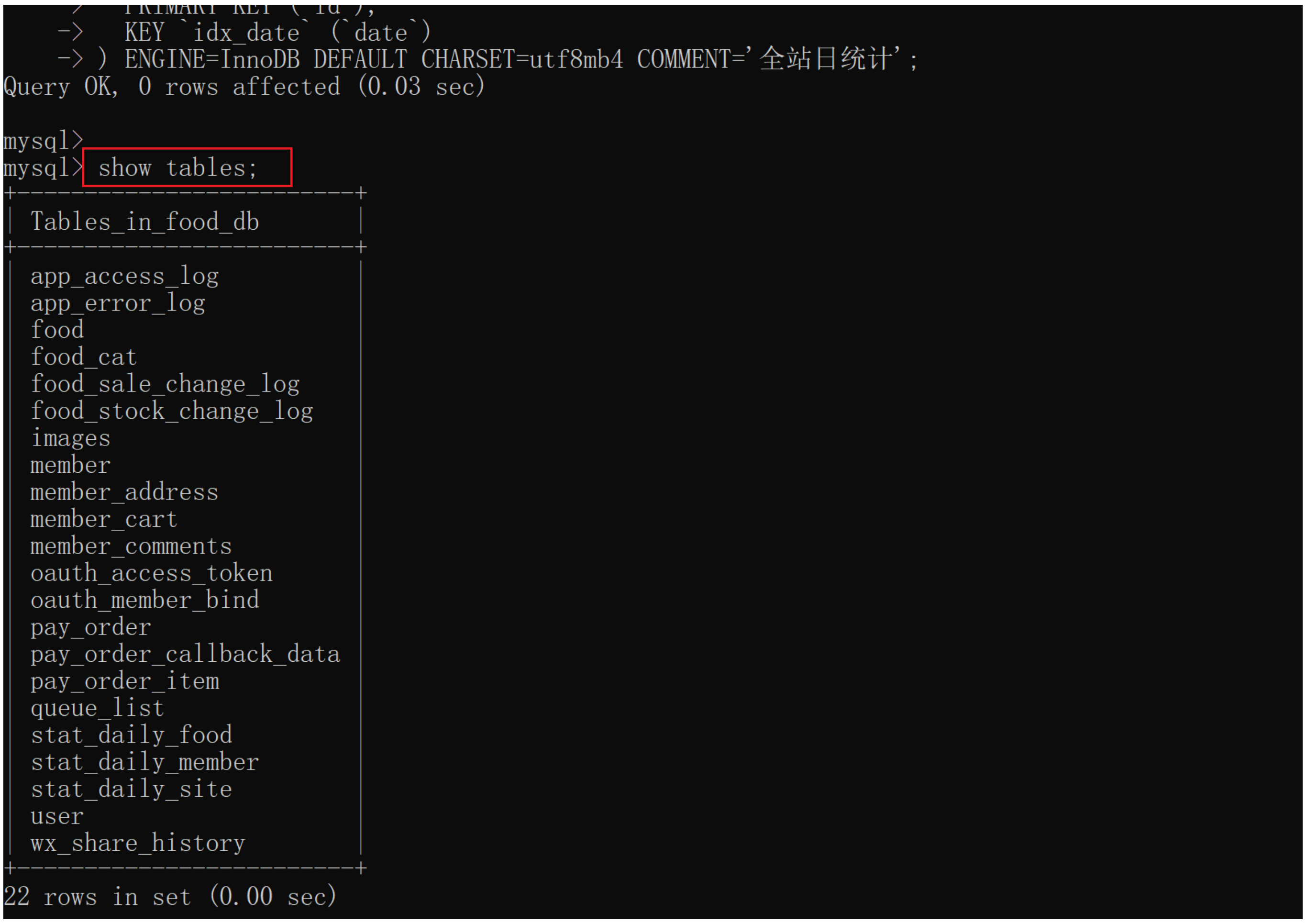


```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| anli               |
| anli2              |
| db_books           |
| demo_book          |
| fldemo             |
| food_db            |
| mysql              |
| performance_schema |
| py                 |
| test               |
| v2ex               |
| xiaodemo           |
| xiaoyu_demo        |
+--------------------+
14 rows in set (0.01 sec)

mysql> use food_db
Database changed
mysql> DROP TABLE IF EXISTS `app_access_log`;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```



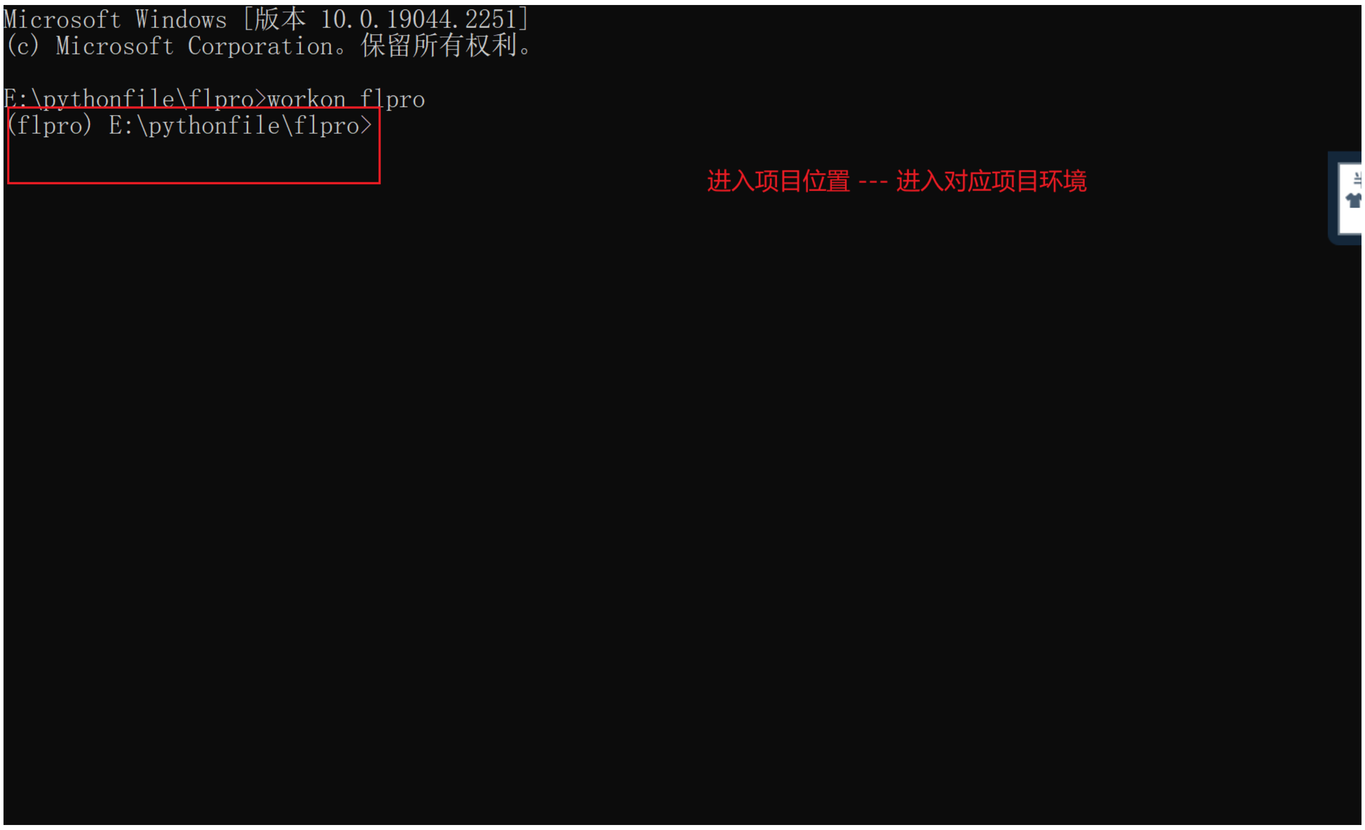
粘贴（根据你自己的东西操作-cmd一般是鼠标右键点一下就是粘贴）

```
14 rows in set (0.01 sec)

mysql> use food_db
Database changed
mysql> DROP TABLE IF EXISTS `app_access_log`;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql>
mysql> CREATE TABLE `app_access_log` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `uid` bigint(20) NOT NULL DEFAULT '0' COMMENT 'uid',
    ->   `referer_url` varchar(255) NOT NULL DEFAULT '' COMMENT '当前访问的refer',
    ->   `target_url` varchar(255) NOT NULL DEFAULT '' COMMENT '访问的url',
    ->   `query_params` text NOT NULL COMMENT 'get和post参数',
    ->   `ua` varchar(255) NOT NULL DEFAULT '' COMMENT '访问ua',
    ->   `ip` varchar(32) NOT NULL DEFAULT '' COMMENT '访问ip',
    ->   `note` varchar(1000) NOT NULL DEFAULT '' COMMENT 'json格式备注字段',
    ->   `created_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    ->   PRIMARY KEY (`id`),
    ->   KEY `idx_uid` (`uid`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='用户访问记录表';
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql>
mysql> DROP TABLE IF EXISTS `app_error_log`;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> CREATE TABLE `app_error_log` (
    ->   `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
    ->   `referer_url` varchar(255) NOT NULL DEFAULT '' COMMENT '当前访问的refer',
    ->   `target_url` varchar(255) NOT NULL DEFAULT '' COMMENT '访问的url',
    ->   `query_params` text NOT NULL COMMENT 'get和post参数',
    ->   `content` longtext NOT NULL COMMENT '日志内容',
```



```
    ->  PRIMARY KEY (`id`),
    ->  KEY `idx_date` (`date`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='全站日统计';
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> show tables;
+--------------------------+
| Tables_in_food_db        |
+--------------------------+
| app_access_log           |
| app_error_log            |
| food                     |
| food_cat                 |
| food_sale_change_log     |
| food_stock_change_log    |
| images                   |
| member                   |
| member_address           |
| member_cart              |
| member_comments          |
| oauth_access_token       |
| oauth_member_bind        |
| pay_order                |
| pay_order_callback_data  |
| pay_order_item           |
| queue_list               |
| stat_daily_food          |
| stat_daily_member        |
| stat_daily_site          |
| user                     |
| wx_share_history         |
+--------------------------+
22 rows in set (0.00 sec)
```

2.模型建立

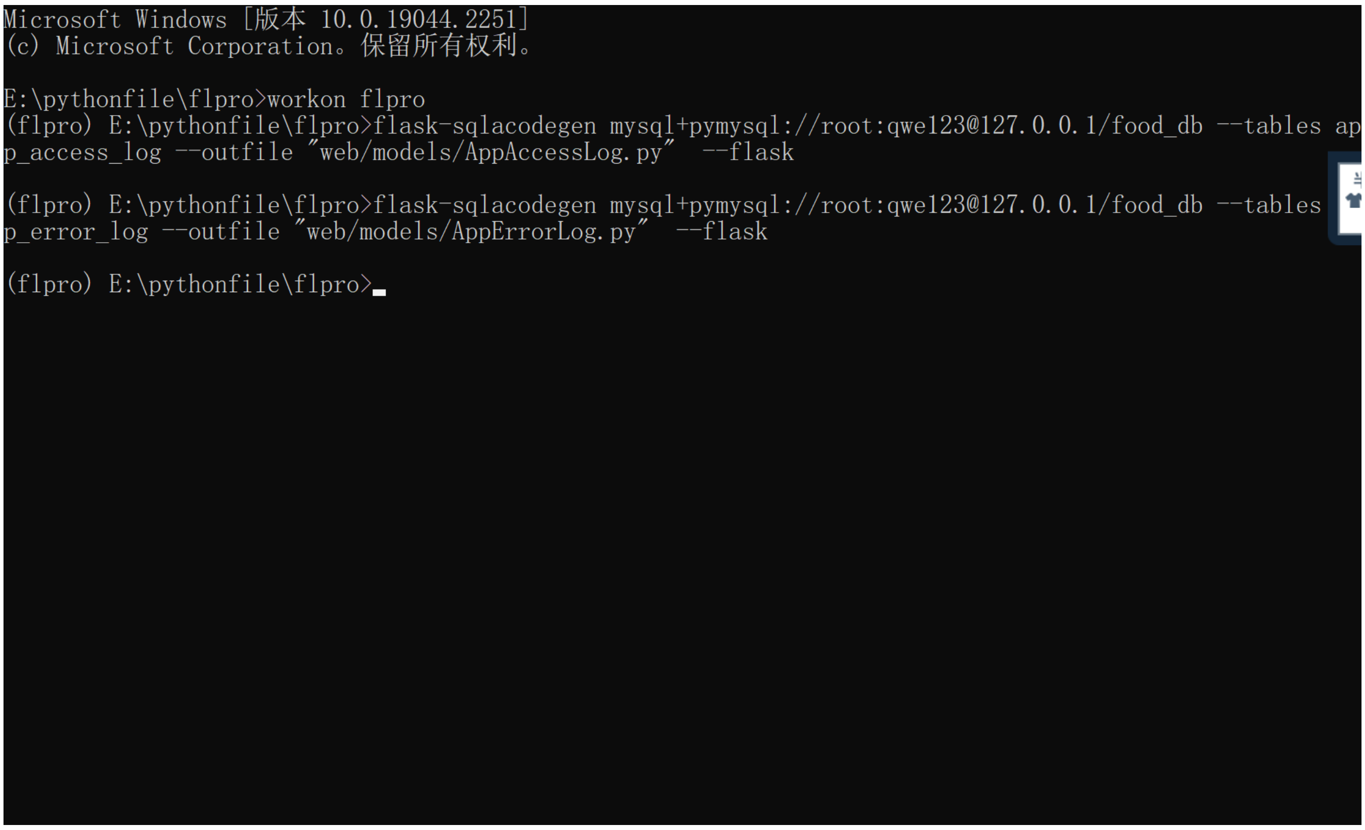


```
Microsoft Windows [版本 10.0.19044.2251]
(c) Microsoft Corporation。保留所有权利。

E:\pythonfile\flpro>workon flpro
(flpro) E:\pythonfile\flpro>
```

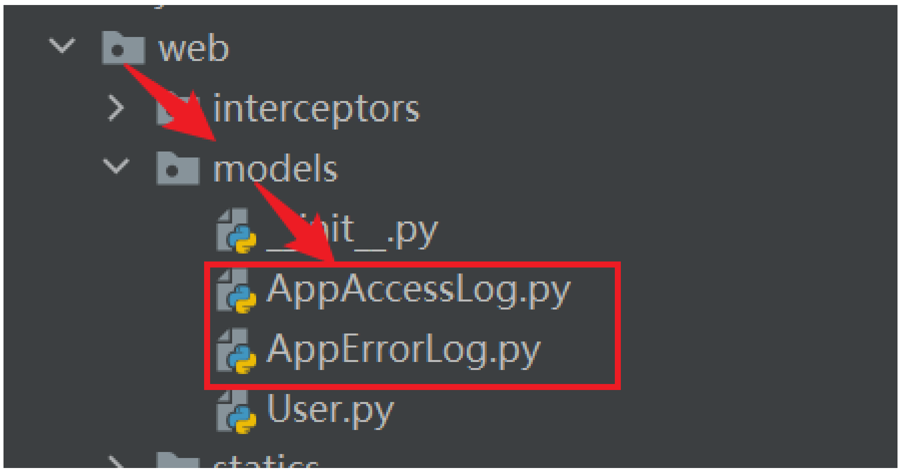
进入项目位置 --- 进入对应项目环境

反推建立模型类 --- 不熟悉的参考第二章节（2.登录部分）

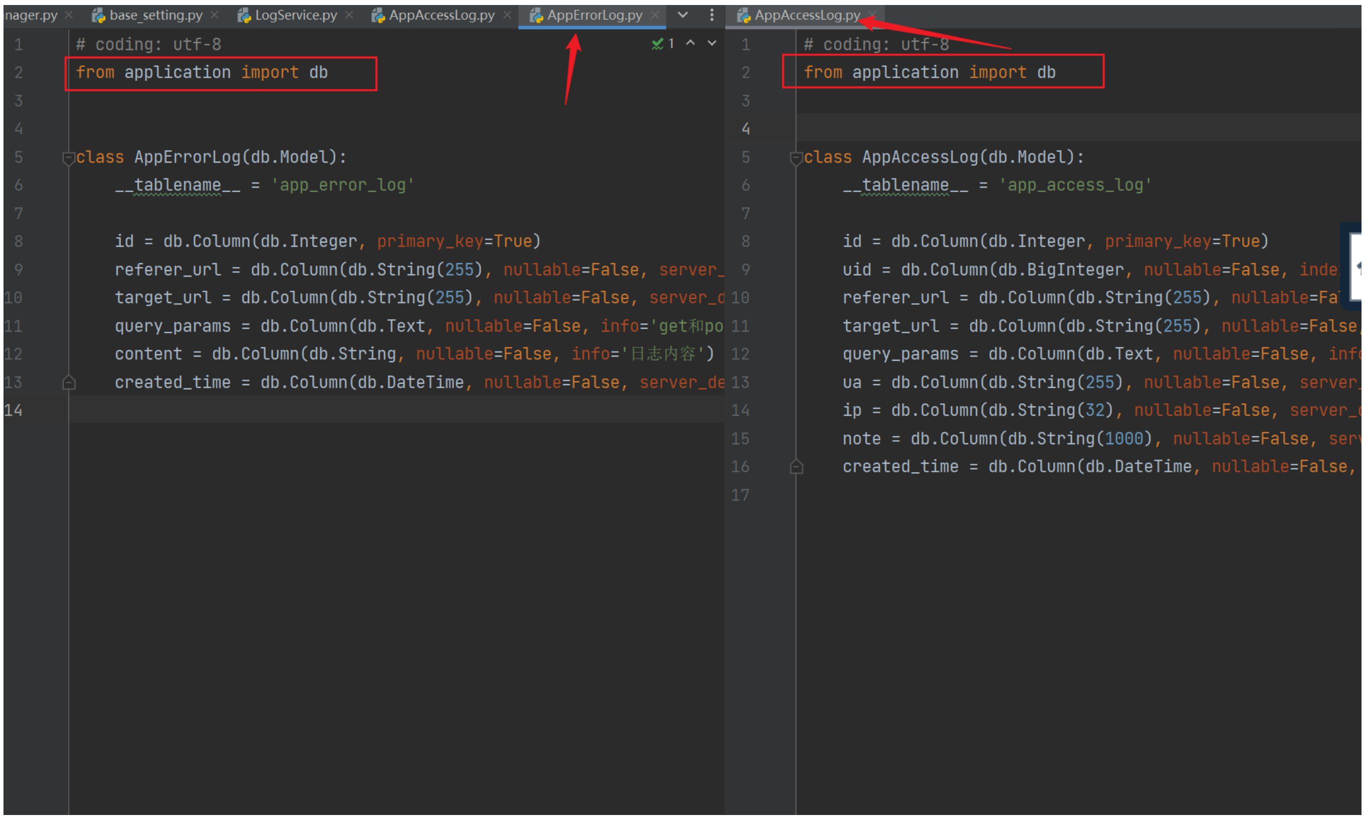
```
flask-sqlacodegen mysql+pymysql://root:qwe123@127.0.0.1/food_db --tables app_access_log --outfile
"web/models/AppAccessLog.py"  --flask

flask-sqlacodegen mysql+pymysql://root:qwe123@127.0.0.1/food_db --tables app_error_log --outfile
"web/models/AppErrorLog.py"  --flask
```



```
Microsoft Windows [版本 10.0.19044.2251]
(c) Microsoft Corporation。保留所有权利。

E:\pythonfile\flpro>workon flpro
(flpro) E:\pythonfile\flpro>flask-sqlacodegen mysql+pymysql://root:qwe123@127.0.0.1/food_db --tables ap
p_access_log --outfile "web/models/AppAccessLog.py"  --flask

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

(flpro) E:\pythonfile\flpro>_
```



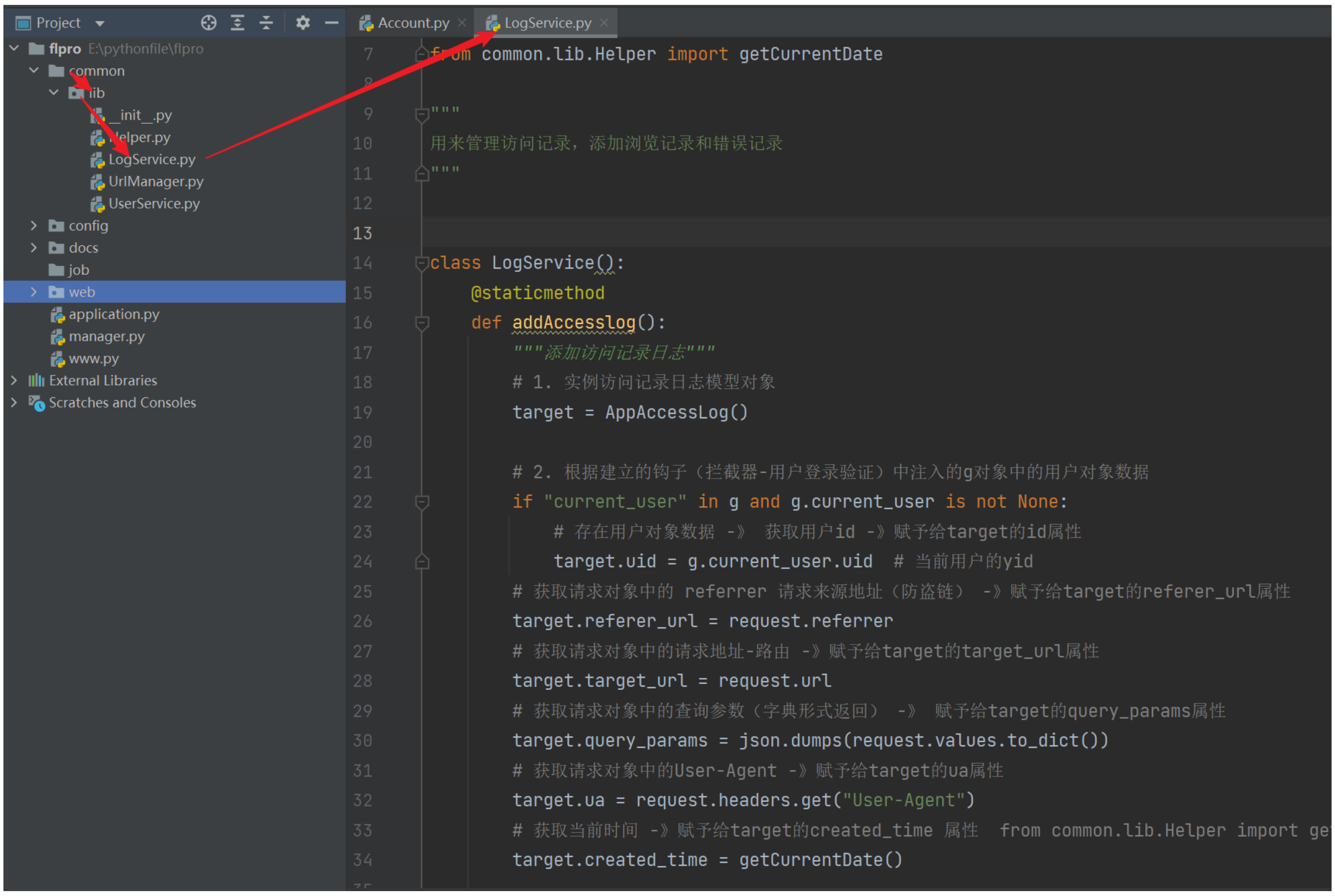
3.调整模型db对象



```python
# coding: utf-8
from application import db


class AppErrorLog(db.Model):
    __tablename__ = 'app_error_log'

    id = db.Column(db.Integer, primary_key=True)
    referer_url = db.Column(db.String(255), nullable=False, server_
    target_url = db.Column(db.String(255), nullable=False, server_d
    query_params = db.Column(db.Text, nullable=False, info='get和po
    content = db.Column(db.String, nullable=False, info='日志内容')
    created_time = db.Column(db.DateTime, nullable=False, server_de
```

```python
# coding: utf-8
from application import db


class AppAccessLog(db.Model):
    __tablename__ = 'app_access_log'

    id = db.Column(db.Integer, primary_key=True)
    uid = db.Column(db.BigInteger, nullable=False, inde
    referer_url = db.Column(db.String(255), nullable=Fa
    target_url = db.Column(db.String(255), nullable=False,
    query_params = db.Column(db.Text, nullable=False, inf
    ua = db.Column(db.String(255), nullable=False, server_
    ip = db.Column(db.String(32), nullable=False, server_
    note = db.Column(db.String(1000), nullable=False, serv
    created_time = db.Column(db.DateTime, nullable=False,
```

4.封装日志记录方法



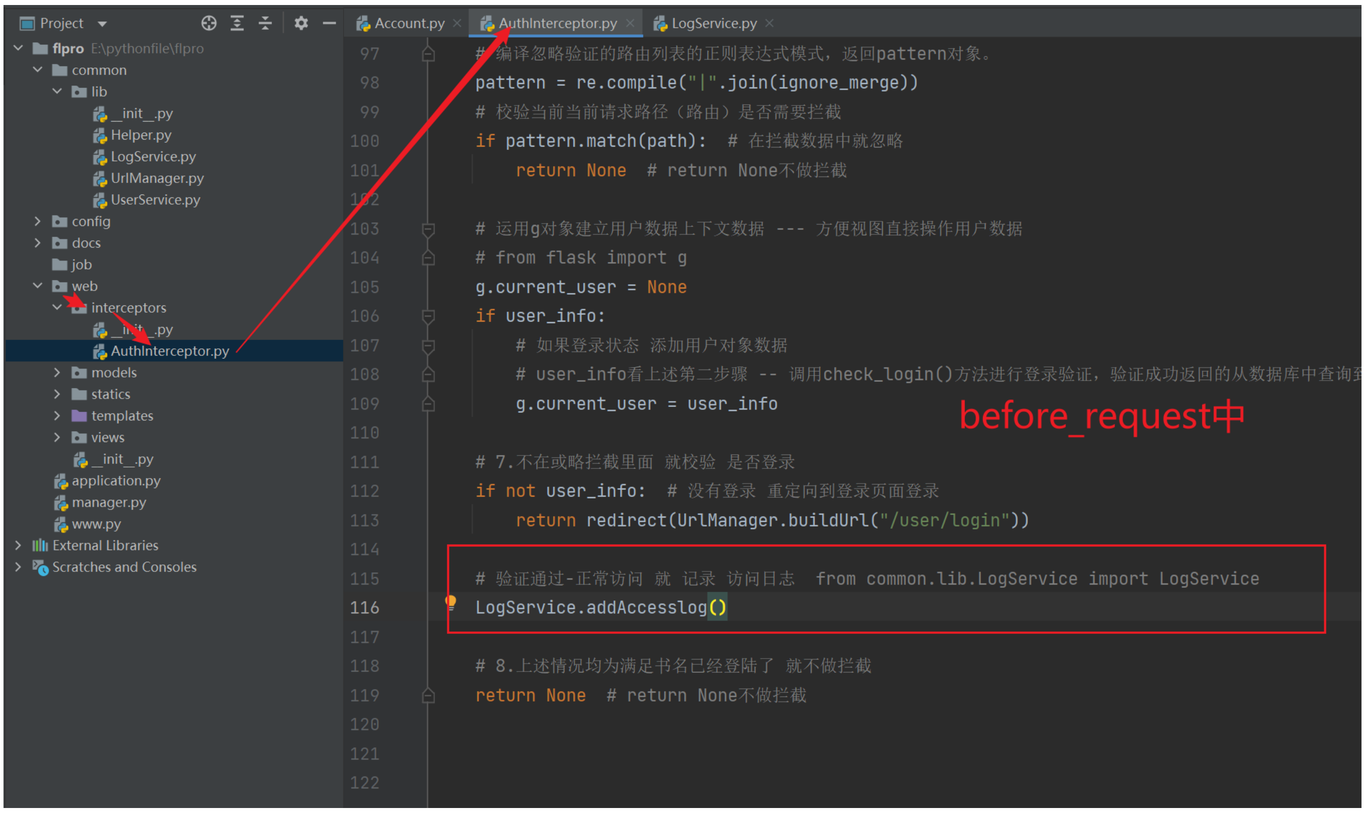
```python
from common.lib.Helper import getCurrentDate

"""
用来管理访问记录，添加浏览记录和错误记录
"""


class LogService():
    @staticmethod
    def addAccesslog():
        """添加访问记录日志"""
        # 1. 实例访问记录日志模型对象
        target = AppAccessLog()

        # 2. 根据建立的钩子（拦截器-用户登录验证）中注入的g对象中的用户对象数据
        if "current_user" in g and g.current_user is not None:
            # 存在用户对象数据 -》 获取用户id -》赋予给target的id属性
            target.uid = g.current_user.uid  # 当前用户的yid
        # 获取请求对象中的 referrer 请求来源地址（防盗链） -》赋予给target的referer_url属性
        target.referer_url = request.referrer
        # 获取请求对象中的请求地址-路由 -》赋予给target的target_url属性
        target.target_url = request.url
        # 获取请求对象中的查询参数（字典形式返回） -》 赋予给target的query_params属性
        target.query_params = json.dumps(request.values.to_dict())
        # 获取请求对象中的User-Agent -》赋予给target的ua属性
        target.ua = request.headers.get("User-Agent")
        # 获取当前时间 -》赋予给target的created_time 属性  from common.lib.Helper import ge
        target.created_time = getCurrentDate()
```

```python
# -*-coding:utf-8-*-
from flask import request, g
from application import db
```

```
import json
from web.models.AppAccessLog import AppAccessLog
from web.models.AppErrorLog import AppErrorLog
from common.lib.Helper import getCurrentDate

"""
用来管理访问记录，添加浏览记录和错误记录
"""


class LogService():
    @staticmethod
    def addAccesslog():
        """添加访问记录日志"""
        # 1. 实例访问记录日志模型对象
        target = AppAccessLog()

        # 2. 根据建立的钩子（拦截器-用户登录验证）中注入的g对象中的用户对象数据
        if "current_user" in g and g.current_user is not None:
            # 存在用户对象数据 -》 获取用户id -》赋予给target的id属性
            target.uid = g.current_user.uid  # 当前用户的yid
        # 获取请求对象中的 referrer 请求来源地址（防盗链） -》赋予给target的referer_url属性
        target.referer_url = request.referrer
        # 获取请求对象中的请求地址-路由 -》赋予给target的target_url属性
        target.target_url = request.url
        # 获取请求对象中的查询参数（字典形式返回） -》 赋予给target的query_params属性
        target.query_params = json.dumps(request.values.to_dict())
        # 获取请求对象中的User-Agent -》赋予给target的ua属性
        target.ua = request.headers.get("User-Agent")
        # 获取当前时间 -》赋予给target的created_time 属性  from common.lib.Helper import
getCurrentDate
        target.created_time = getCurrentDate()

        # 写入数据库
        db.session.add(target)
        db.session.commit()

        return True


    @staticmethod
    def addErrorlog(content):
        """
            添加错误日志
        :param content: 错误信息
        :return:
        """

        # 忽略favicon.ico请求引发的错误
        if "favicon.ico" in request.url:
            print("图标问题报404可以忽略")
            return
        # 实例错误日志模型对象
        target = AppErrorLog()

        # 获取请求对象中的 referrer 请求来源地址（防盗链） -》赋予给target的referer_url属性
        target.referer_url = request.referrer
        # 获取请求对象中的请求地址-路由 -》赋予给target的target_url属性
        target.target_url = request.url
        # 获取请求对象中的查询参数（字典形式返回） -》 赋予给target的query_params属性
        target.query_params = json.dumps(request.values.to_dict())
        # 错误信息内容 -》 赋予给target的content属性
        target.content = content
        # 获取当前时间 -》赋予给target的created_time 属性  from common.lib.Helper import
getCurrentDate
        target.created_time = getCurrentDate()

        # 写入数据库
        db.session.add(target)
        db.session.commit()
        return True
```

5.日志记录插入

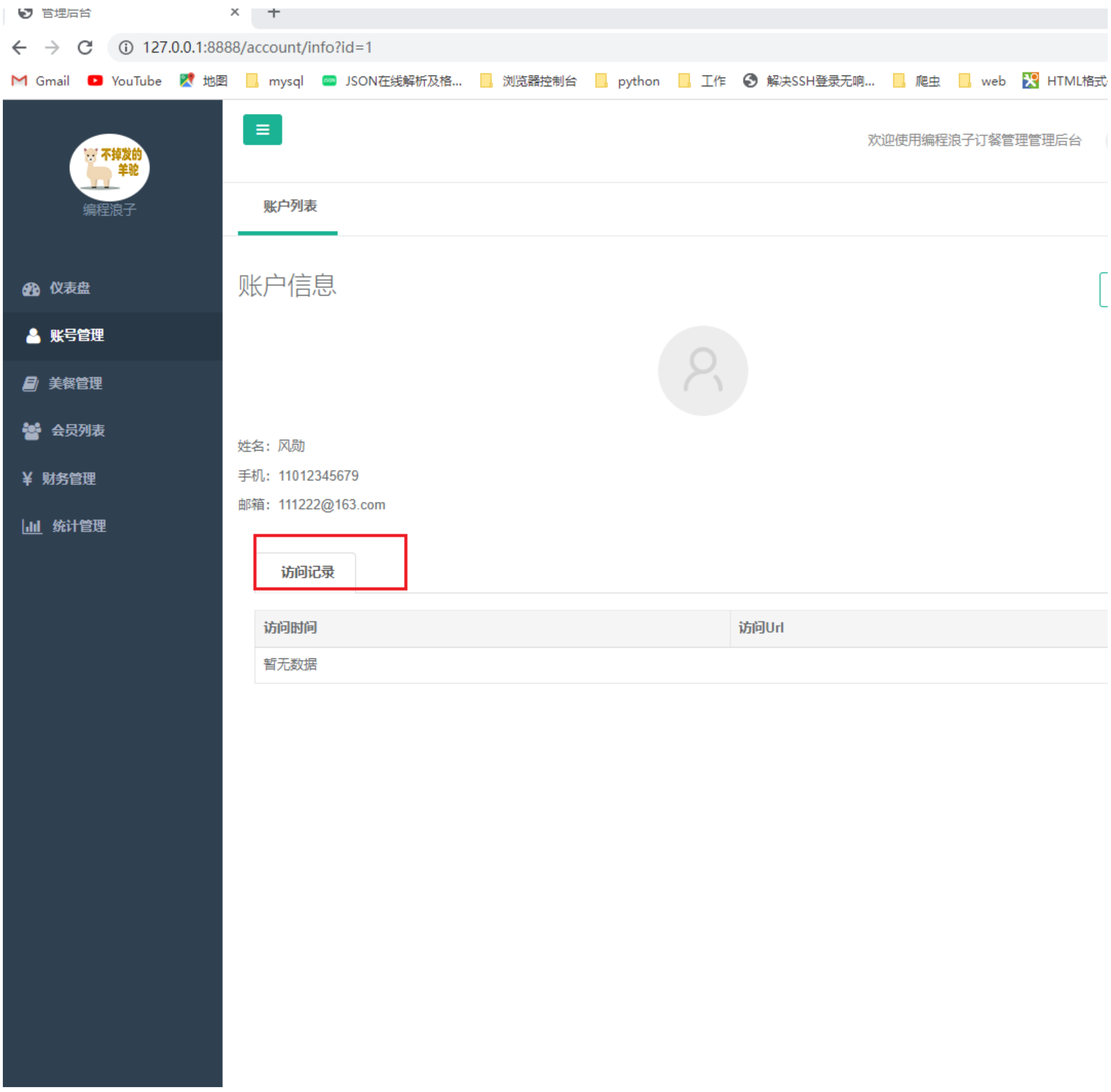
访问日志



```
# 编译忽略验证的路由列表的正则表达式模式，返回pattern对象。
pattern = re.compile("|".join(ignore_merge))
# 校验当前当前请求路径（路由）是否需要拦截
if pattern.match(path):  # 在拦截数据中就忽略
    return None  # return None不做拦截

# 运用g对象建立用户数据上下文数据 --- 方便视图直接操作用户数据
# from flask import g
g.current_user = None
if user_info:
    # 如果登录状态 添加用户对象数据
    # user_info看上述第二步骤 -- 调用check_login()方法进行登录验证，验证成功返回的从数据库中查询
    g.current_user = user_info

# 7.不在或略拦截里面 就校验 是否登录
if not user_info:  # 没有登录 重定向到登录页面登录
    return redirect(UrlManager.buildUrl("/user/login"))

# 验证通过-正常访问 就 记录 访问日志  from common.lib.LogService import LogService
LogService.addAccesslog()

# 8.上述情况均为满足书名已经登陆了 就不做拦截
return None  # return None不做拦截
```

6.日志记录展示

首先你得知道 日志信息在哪展示的 -- http://127.0.0.1:8888/account/info?id=1 --- 也就用户详情页中展示

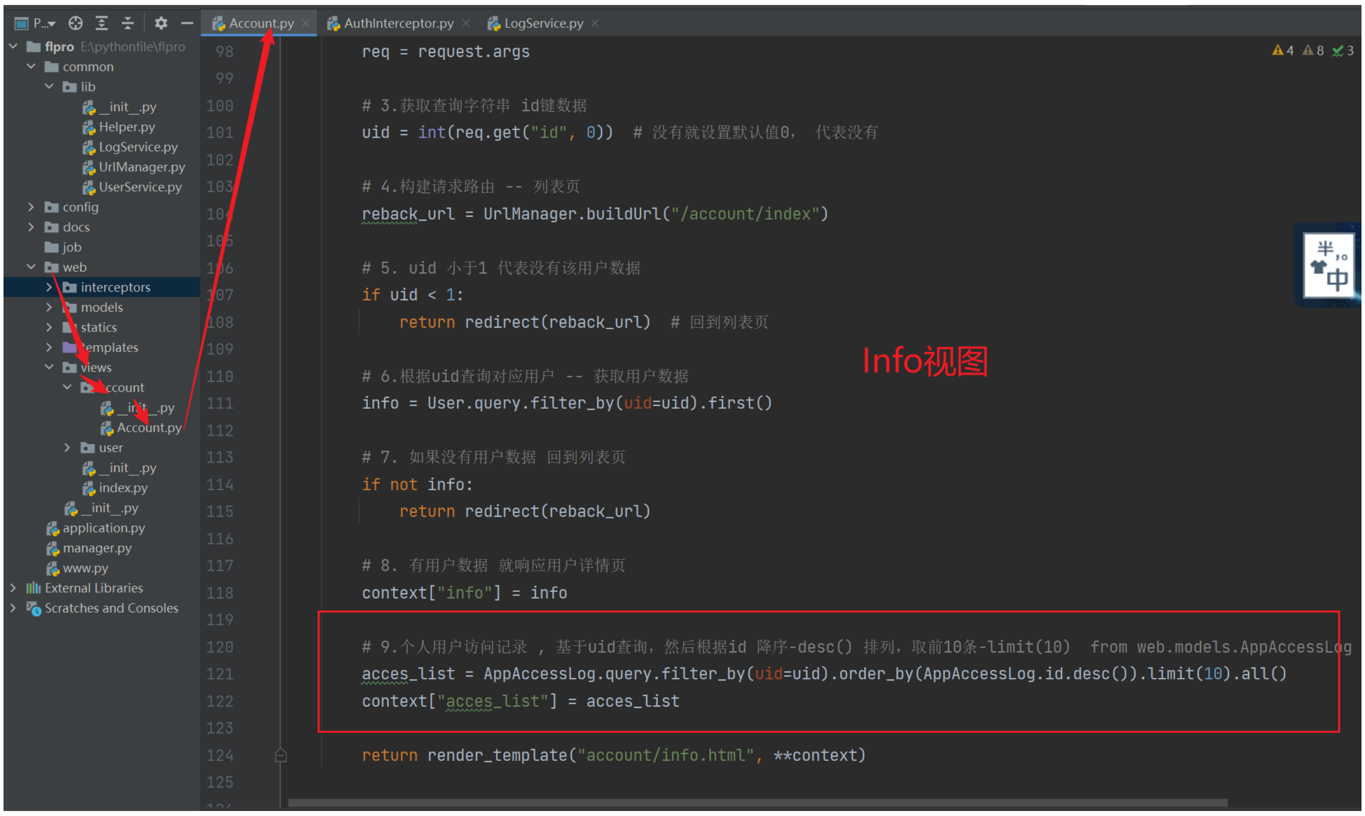


账户信息

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

访问时间	访问Url
暂无数据	

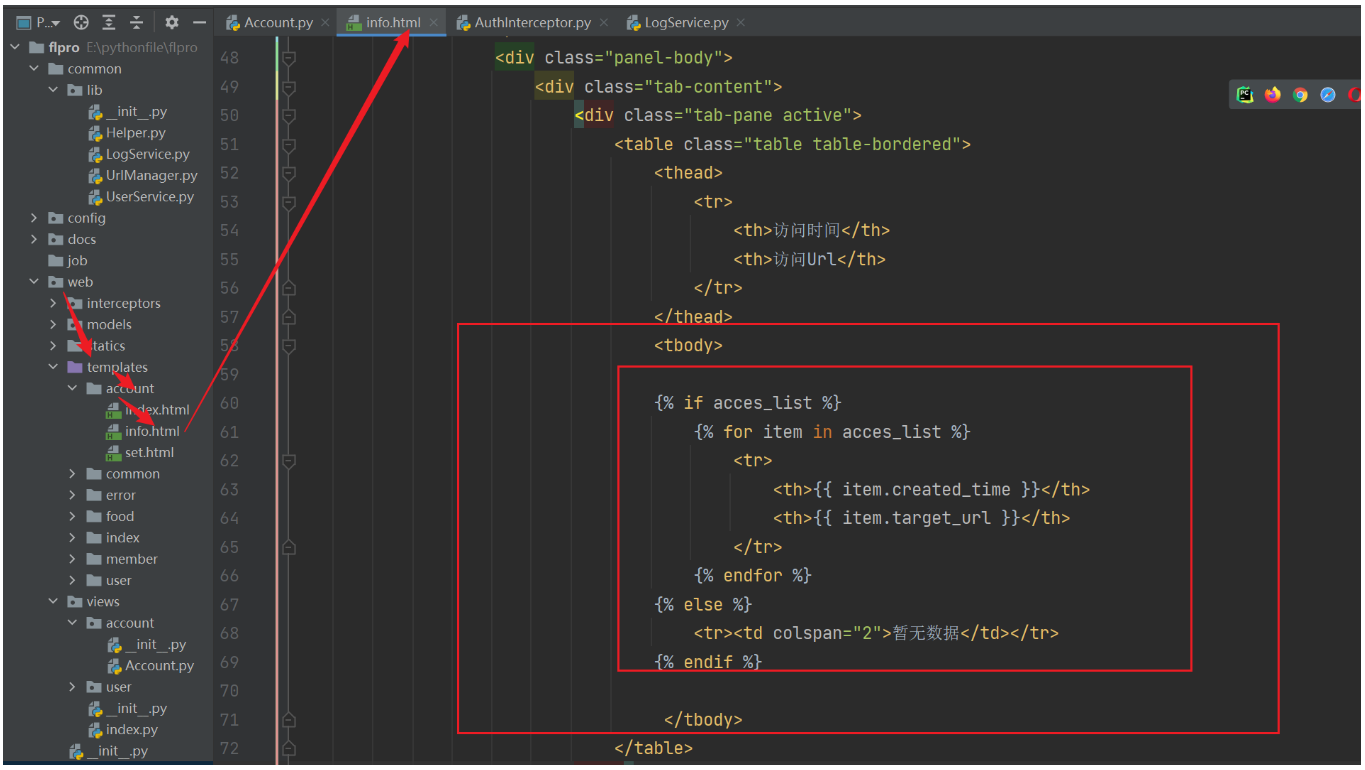
然后就可以进行视图操作了



```
        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

        # 9.个人用户访问记录 ， 基于uid查询，然后根据id 降序-desc() 排列，取前10条-limit(10)  from web.models.AppAccessLog
        acces_list = AppAccessLog.query.filter_by(uid=uid).order_by(AppAccessLog.id.desc()).limit(10).all()
        context["acces_list"] = acces_list

        return render_template("account/info.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

        # 9.个人用户访问记录 ， 基于uid查询，然后根据id 降序-desc() 排列，取前10条-limit(10)  from 
web.models.AppAccessLog import AppAccessLog
        acces_list = 
AppAccessLog.query.filter_by(uid=uid).order_by(AppAccessLog.id.desc()).limit(10).all()
        context["acces_list"] = acces_list
        return render_template("account/info.html", **context)
```

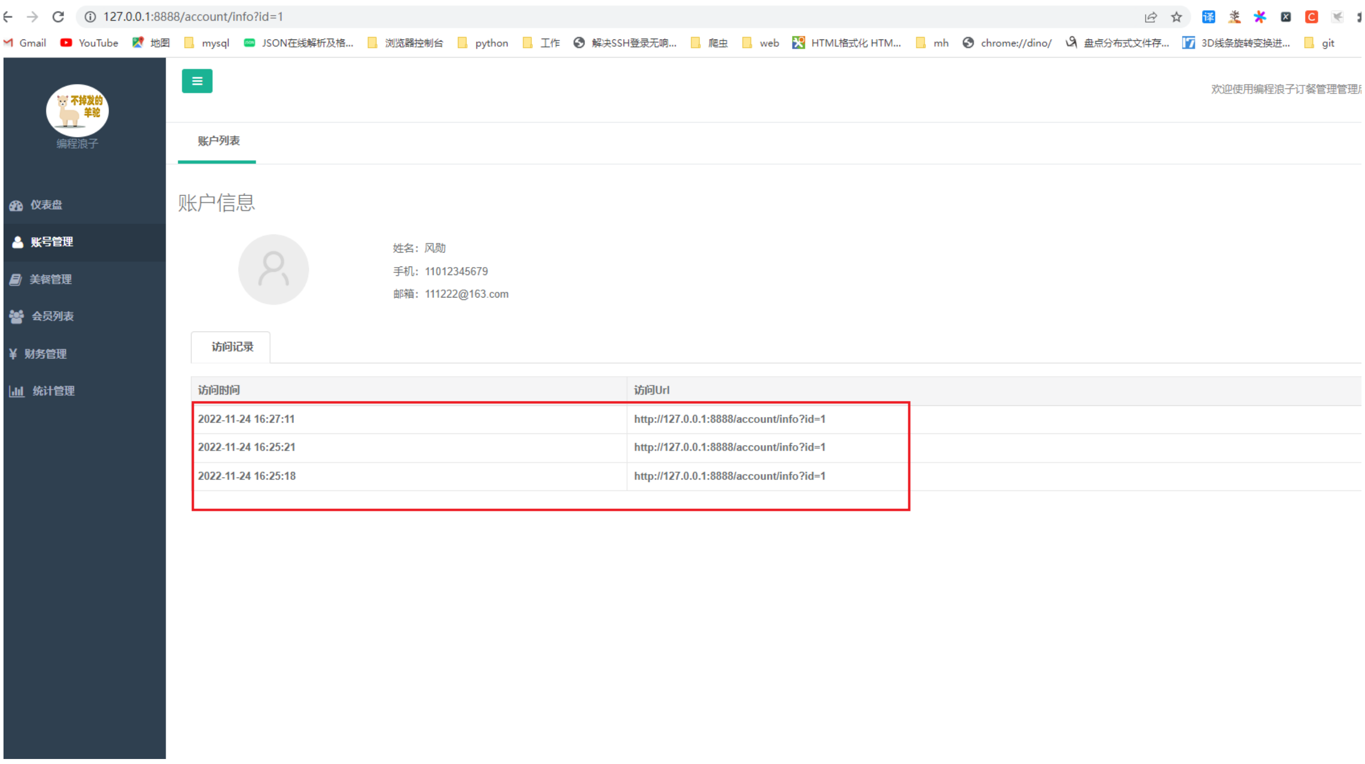
视图数据渲染后 -- 调整模板



```
<div class="panel-body">
    <div class="tab-content">
        <div class="tab-pane active">
            <table class="table table-bordered">
                <thead>
                    <tr>
                        <th>访问时间</th>
                        <th>访问Url</th>
                    </tr>
                </thead>
                <tbody>

                {% if acces_list %}
                    {% for item in acces_list %}
                        <tr>
                            <th>{{ item.created_time }}</th>
                            <th>{{ item.target_url }}</th>
                        </tr>
                    {% endfor %}
                {% else %}
                    <tr><td colspan="2">暂无数据</td></tr>
                {% endif %}

                 </tbody>
            </table>
```

7.启动测试

http://127.0.0.1:8888/account/info?id=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
2022-11-24 16:27:11	http://127.0.0.1:8888/account/info?id=1
2022-11-24 16:25:21	http://127.0.0.1:8888/account/info?id=1
2022-11-24 16:25:18	http://127.0.0.1:8888/account/info?id=1