

1.环境搭建

接触过虚拟环境的同学可通过虚拟环境

未接触的同学可使用你自己的主环境

虚拟环境创建

接触过虚拟环境的同学可使用此操作

```
mkvirtualenv flpro # 创建flpro环境
```

```
C:\Users\denhu\mkvirtualenv flpro
created virtual environment CPython3.7.7.final.0-64 in 4478ms
  creator CPython3Windows(dest=D:\EVNS\flpro, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\denhu\AppData\Local\ppya\virtualenv)
    added seed packages: pip==22.3, setuptools==65.5.0, wheel==0.37.1
  activators BashActivator, BatchActivator, FishActivator, NushellActivator, PowerShellActivator, PythonActivator
(flpro) C:\Users\denhu>
```

安装flask

```
pip install flask
```

```
(flpro) C:\Users\denhu>pip install flask
Collecting flask
  Using cached Flask-2.2.2-py3-none-any.whl (101 kB)
Collecting click>=8.0
  Using cached click-8.1.3-py3-none-any.whl (96 kB)
Collecting importlib-metadata>=3.6.0
  Using cached importlib_metadata-5.0.0-py3-none-any.whl (21 kB)
Collecting itsdangerous>=2.0
  Using cached itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2>=3.0
  Using cached Jinja2-3.1.2-py3-none-any.whl (133 kB)
Collecting Werkzeug>=2.2.2
  Using cached Werkzeug-2.2.2-py3-none-any.whl (232 kB)
Collecting colorama
  Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting typing_extensions>=3.6.4
  Using cached typing_extensions-4.4.0-py3-none-any.whl (26 kB)
Collecting zipp>=0.5
  Using cached zipp-3.10.0-py3-none-any.whl (6.2 kB)
Collecting MarkupSafe>=2.0
  Using cached MarkupSafe-2.1.1-cp37-cp37m-win_amd64.whl (17 kB)
Installing collected packages: zipp, typing_extensions, MarkupSafe, itsdangerous, colorama, Werkzeug, Jinja2, importlib-metadata, click, flask
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.1 Werkzeug-2.2.2 click-8.1.3 colorama-0.4.6 flask-2.2 importlib-metadata-5.0.0 itsdangerous-2.1.2 typing_extensions-4.4.0 zipp-3.10.0
```

```
(flpro) C:\Users\denhu>pip list
```

Package	Version
click	8.1.3
colorama	0.4.6
Flask	2.2.2
importlib-metadata	5.0.0
itsdangerous	2.1.2
Jinja2	3.1.2
MarkupSafe	2.1.1
pip	22.3
setuptools	65.5.0
typing_extensions	4.4.0
Werkzeug	2.2.2
wheel	0.37.1
zipp	3.10.0

2.项目创建

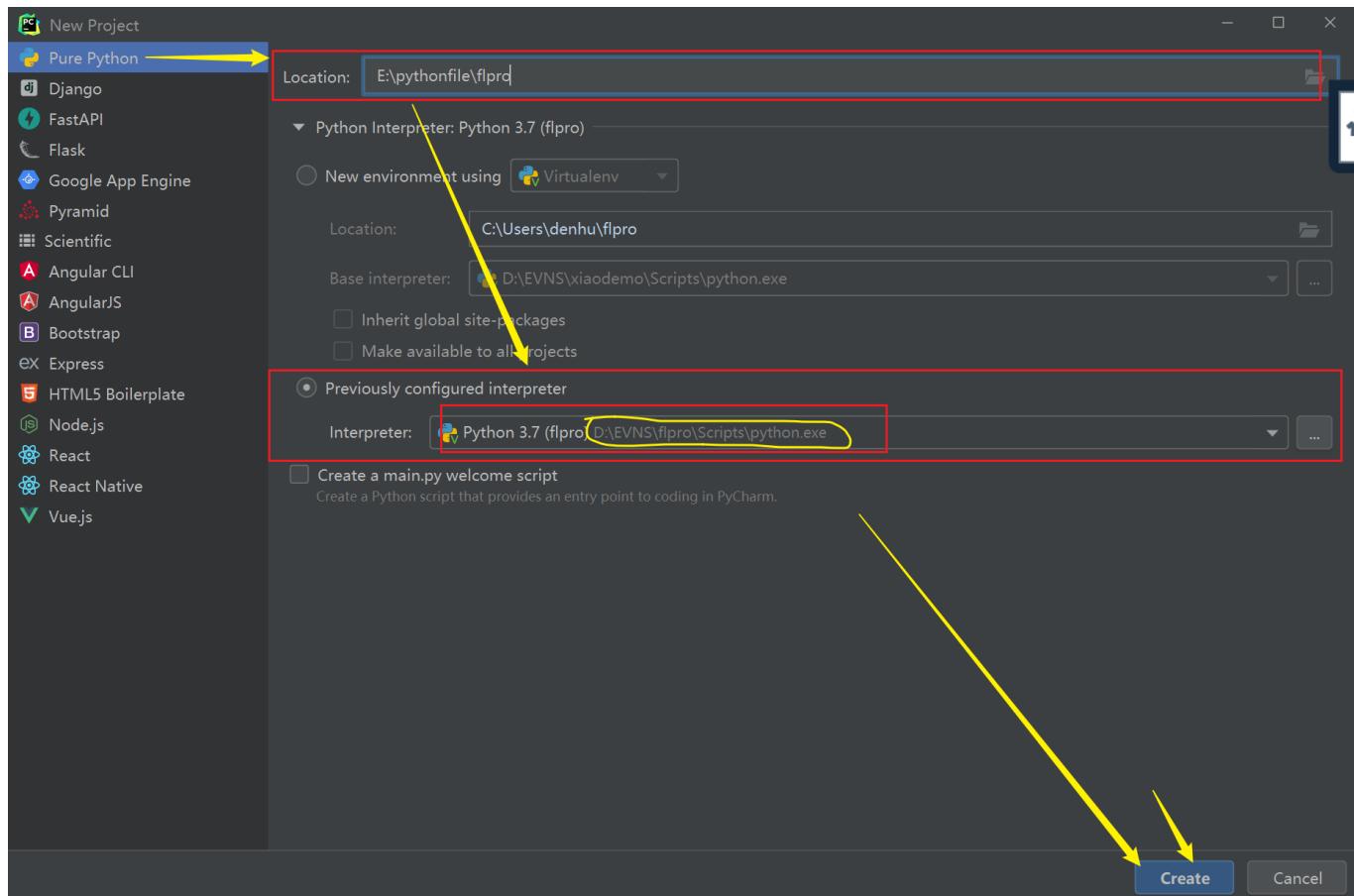
查看环境位置

```
where python
```

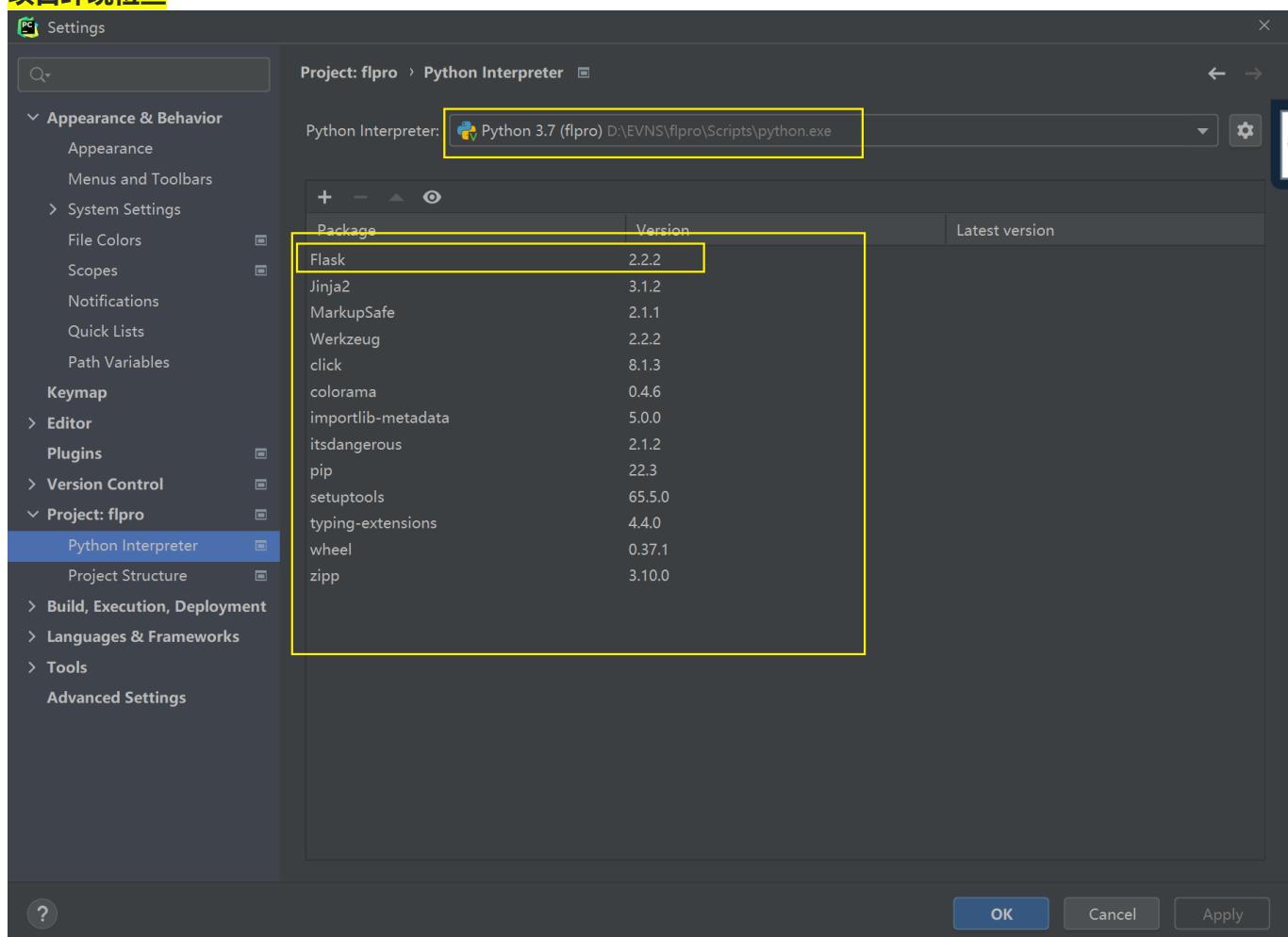
```
(flpro) C:\Users\denhu>where python
D:\EVNS\f1pro\Scripts\python.exe
D:\应用\python37\python.exe
```

```
(flpro) C:\Users\denhu>
```

创建项目



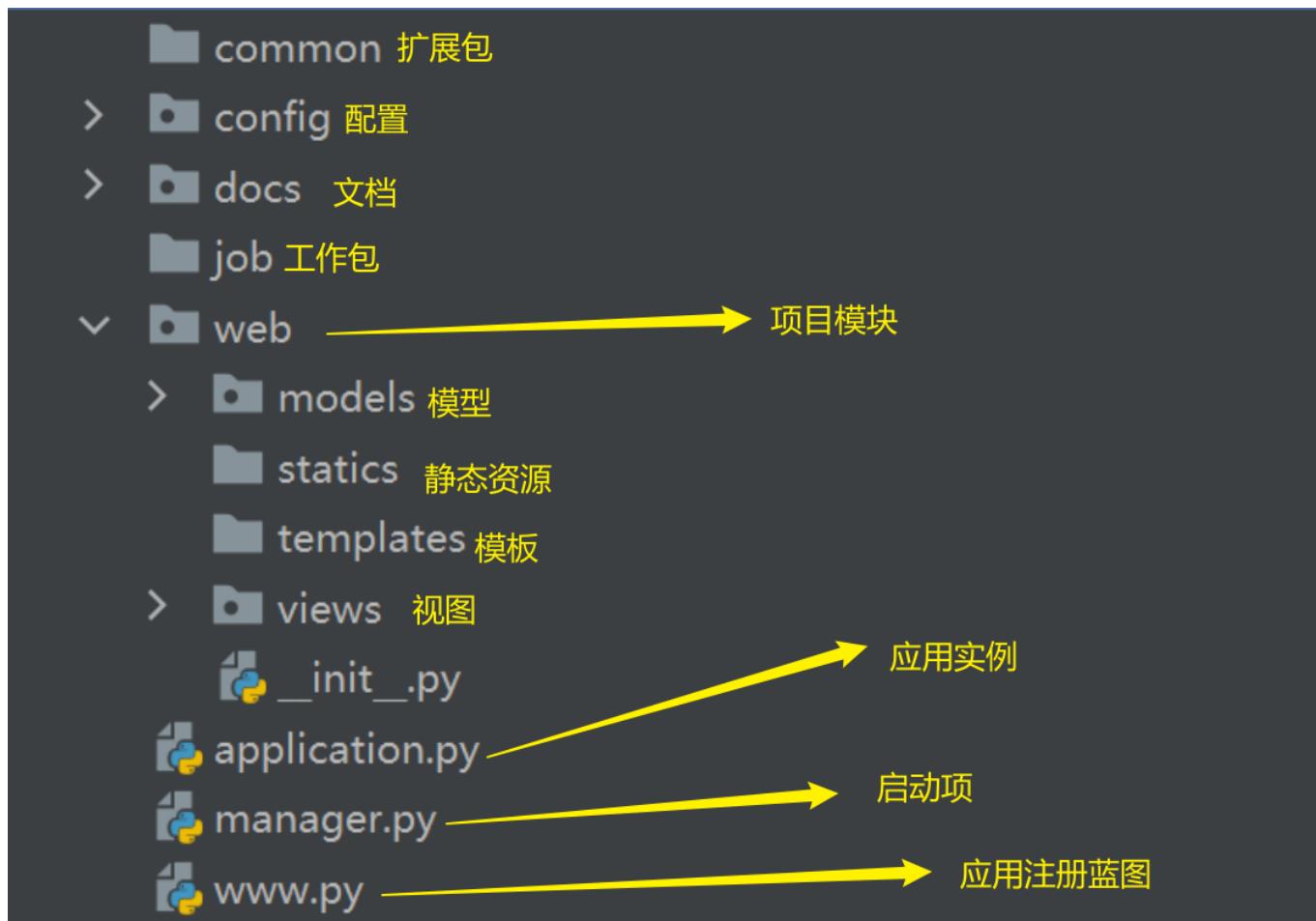
项目环境检查



3.项目布局

web模块中仅包含models, statics, templates, views

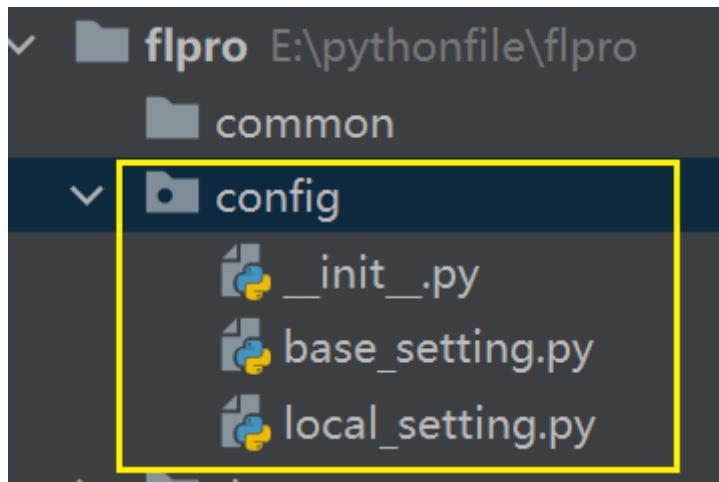
注意辨别同级问题



4.配置编写

创建mysql库

```
create database food_db charset=utf8;
```



```
# -*- encoding: utf-8 -*-
"""
File      : config/base_setting.py
teaching   :
    flask配置
"""

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

```
# -*- encoding: utf-8 -*-
"""
File      : config/local_setting.py
teaching   :
    数据库连接配置
"""

DEBUG = True
SQLALCHEMY_ECHO = True
SQLALCHEMY_DATABASE_URI = 'mysql+pymysql://root:qwe123@127.0.0.1/food_db?charset=utf8'
SQLALCHEMY_TRACK_MODIFICATIONS = False
SQLALCHEMY_ENCODING = "utf8"
```

5.项目搭建-搭建应用实例

```
# 安装 Flask-SQLAlchemy
pip install Flask-SQLAlchemy -i https://pypi.tuna.tsinghua.edu.cn/simple

# 安装pymysql
pip install pymysql -i https://pypi.tuna.tsinghua.edu.cn/simple
```

```
# 安装flask_script
pip install flask-script -i https://pypi.tuna.tsinghua.edu.cn/simple
```

扩展

```
# 1.platform模块
import platform
print(platform.system().lower()) # 获取系统类型 例如: windows
```

application.py

```
# -*- encoding: utf-8 -*-
"""
File      : application.py
teaching   :
    应用实例
"""

from flask import Flask
from flask_script import Manager
from flask_sqlalchemy import SQLAlchemy
import os
import platform
from common.lib.UrlManager import UrlManager

# 实例数据库对象
db = SQLAlchemy()

# 重构flask初始化
class Application(Flask):

    def __init__(self, import_name, template_folder=None, root_path=None, static_folder='statics'):
        """
        :param import_name:
        :param template_folder: 模板文件
        :param root_path: 应用程序文件的根路径
        """

        # 继承原有初始化
        super(Application, self).__init__(import_name,
                                           template_folder=template_folder,
                                           static_folder=static_folder,
                                           root_path=root_path)

        # 加载Flask配置
        self.config.from_pyfile('config/base_setting.py') # 不管本地还是线上都需要的配置

        # 根据系统选择配置加载
        if platform.system().lower() == "windows":
            self.config.from_pyfile('config/local_setting.py') # 本地配置
        elif platform.system().lower() == "linux":
            self.config.from_pyfile('config/production_setting.py') # 可在config下建立一个
production_setting文件
        else: # 通用配置
```

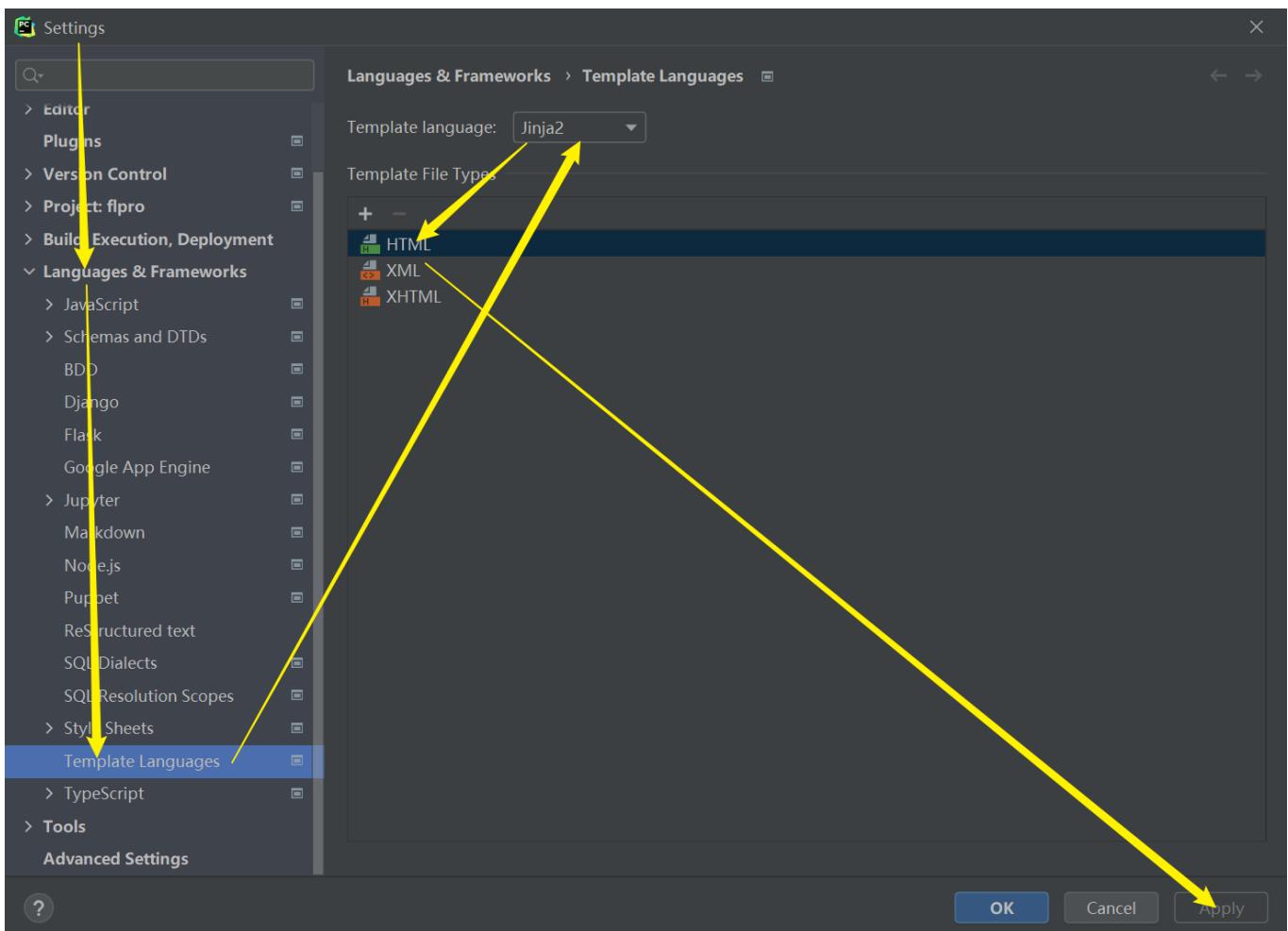
```
self.config.from_pyfile('config/local_setting.py')

# 初始化数据库对象连接配置
db.init_app(self)

# 拼接模板目录绝对路径 --- 适应项目上线, 测试等移植部署
# os.getcwd() 获取当前文件 所在目录的绝对路径
template_folder = os.getcwd() + "/web/templates/"
# print('template_folder:', template_folder)

# 实例应用对象
app = Application(__name__, template_folder=template_folder, root_path=os.getcwd())

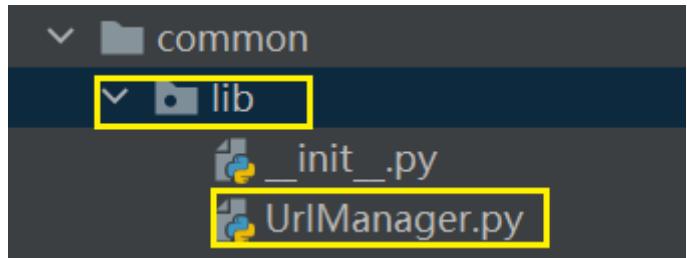
# 实例应用管理对象
manager = Manager(app)
```



全局模板函数

在common模块中创建lib目录

在common的lib目录中建立UrlManager.py文件



```
# -*- coding: utf-8 -*-
# common/lib/UrlManager.py
class UrlManager(object):
    def __init__(self):
        pass

    @staticmethod
    def buildUrl(path):
        """
        构建地址
        :param path:
        :return: 地址
        """
        return path

    @staticmethod
    def buildStaticUrl(path):
        """
        构建静态地址
        :param path:
        :return: 地址
        """
        ver = "%s%(22222222"
        path = "/static" + path + "?ver=" + ver
        return UrlManager.buildUrl(path)
```

回到application.py，进行注册全局模板函数

```
application.py  UrlManager.py
57 app = Application(__name__, template_folder=template_folder, root_path=os.getcwd())
58
59 # 实例应用管理对象
60 manager = Manager(app)
61
62 # 加载自定义全局模板函数
63 # from common.libs.UrlManager import UrlManager
64 app.add_template_global(UrlManager.buildStaticUrl,"buildStaticUrl")
65 app.add_template_global(UrlManager.buildUrl,"buildUrl")
66
67 """
68 函数模板
69
70 add_template_global 注册一个自定义全局函数。与'template_global'装饰器的工作原理完全相同。
71 其中第一个参数是自定义的全局函数，第二个参数是自定义的全局函数的名字
72
73 def bar():
74     return "I am bar."
75 app.add_template_global(bar, 'bar')
76
77 模板中使用：
78 <p>调用bar() 的结果为：{{ bar() }}</p>
79 """

```

```
# 加载自定义全局模板函数
# from common.lib.UrlManager import UrlManager
app.add_template_global(UrlManager.buildStaticUrl,"buildStaticUrl")
app.add_template_global(UrlManager.buildUrl,"buildUrl")
```

....

函数模板

```
add_template_global 注册一个自定义全局函数。与'template_global'装饰器的工作原理完全相同。
其中第一个参数是自定义的全局函数，第二个参数是自定义的全局函数的名字
```

```
def bar():
    return "I am bar."
app.add_template_global(bar, 'bar')
```

模板中使用：

```
<p>调用bar() 的结果为：{{ bar() }}</p>
```

....

6.项目搭建--建立启动项

```
# -*- encoding: utf-8 -*-
"""
File      : manager.py
teaching  :
    启动项
"""

from application import app, manager
from flask_script import Server
```

```
import www

# web server --- Server 运行Flask开发服务器, 即app.run()

# manager.add_command("runserver", Server(host='0.0.0.0',
#                                         port=app.config['SERVER_PORT'],
#                                         use_debugger=True,
#                                         use_reloader=True
#                                         )
# )

# 我们在本地运行 host使用127.0.0.1
manager.add_command("runserver",
                    Server(host='127.0.0.1',
                           port=app.config['SERVER_PORT'],
                           use_debugger=True,
                           use_reloader=True
                           )
                    )

def main(): # 启动函数
    print(app.url_map) # 查看url列表
    manager.run() # 启动

if __name__ == '__main__':
    try:
        import sys
        sys.exit(main()) # main启动运行存在异常 sys.exit终止项目运行
        # sys.exit 通过引发SystemExit异常来退出Python程序
    except Exception as e:
        # 捕捉异常后, 直接反馈异常回溯信息
        import traceback
        traceback.print_exc()

        # print(e)打印结果为:
        #         division by zero, 只知道是报了这个错, 但是却不知道在哪个文件哪个函数哪一行报的错。
        # 也就是说无回溯信息
        #
        # traceback.print_exc()打印结果为: --- 存在完整的回溯信息
        #             Traceback (most recent call last):
        #             File "E:/PycharmProjects/testProject2022/pythonBasic/dataBase/aa.py", line 4, in
<module>
        #             1/0
        # ZeroDivisionError: division by zero
```

运行manager文件， 出现如下问题

```

Traceback (most recent call last):
  File "E:/pythonfile/flpro/manager.py", line 8, in <module>
    from application import app, manager
  File "E:/pythonfile/flpro/application.py", line 10, in <module>
    from flask_script import Manager
  File "D:\EVNS\flpro\lib\site-packages\flask_script\__init__.py", line 15, in <module>
    from flask._compat import text_type
ModuleNotFoundError: No module named 'flask._compat'

```

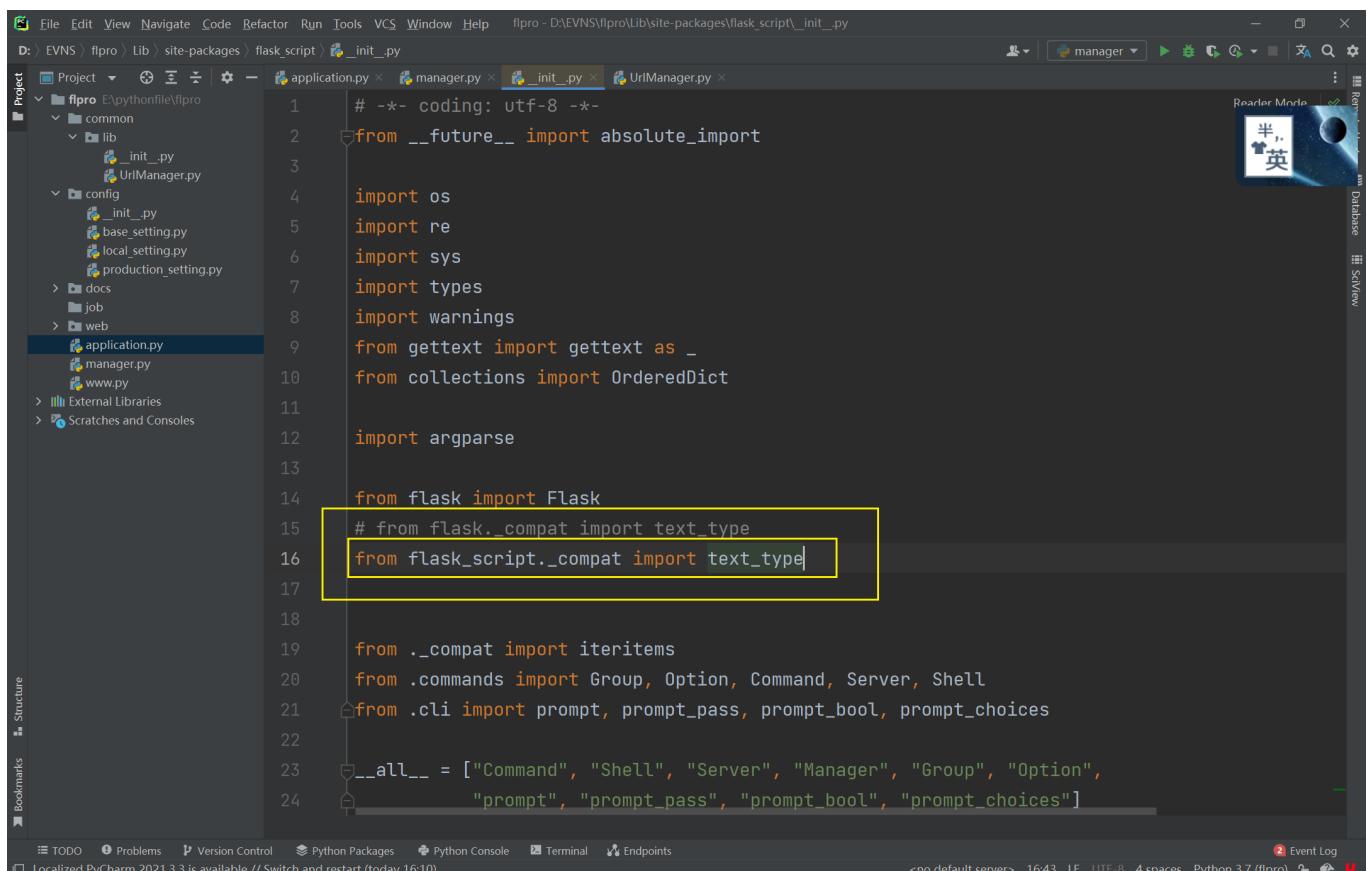
不要慌，这是flask_script版本迭代问题，我们之前讲过
解决方案如下

```

D:\EVNS\flpro\Scripts\python.exe E:/pythonfile/flpro/manager.py
Traceback (most recent call last):
  File "E:/pythonfile/flpro/manager.py", line 8, in <module>
    from application import app, manager
  File "E:/pythonfile/flpro/application.py", line 10, in <module>
    from flask_script import Manager
  File "D:\EVNS\flpro\lib\site-packages\flask_script\__init__.py", line 15, in <module>
    from flask._compat import text_type
ModuleNotFoundError: No module named 'flask._compat'          点击或者根据此路径寻找到目标文件，文本打开

Process finished with exit code 1

```



The screenshot shows the PyCharm IDE interface. The code editor displays the file `__init__.py` with the following content:

```

# -*- coding: utf-8 -*-
from __future__ import absolute_import

import os
import re
import sys
import types
import warnings
from gettext import gettext as _
from collections import OrderedDict

import argparse

from flask import Flask
# from flask._compat import text_type
from flask_script._compat import text_type

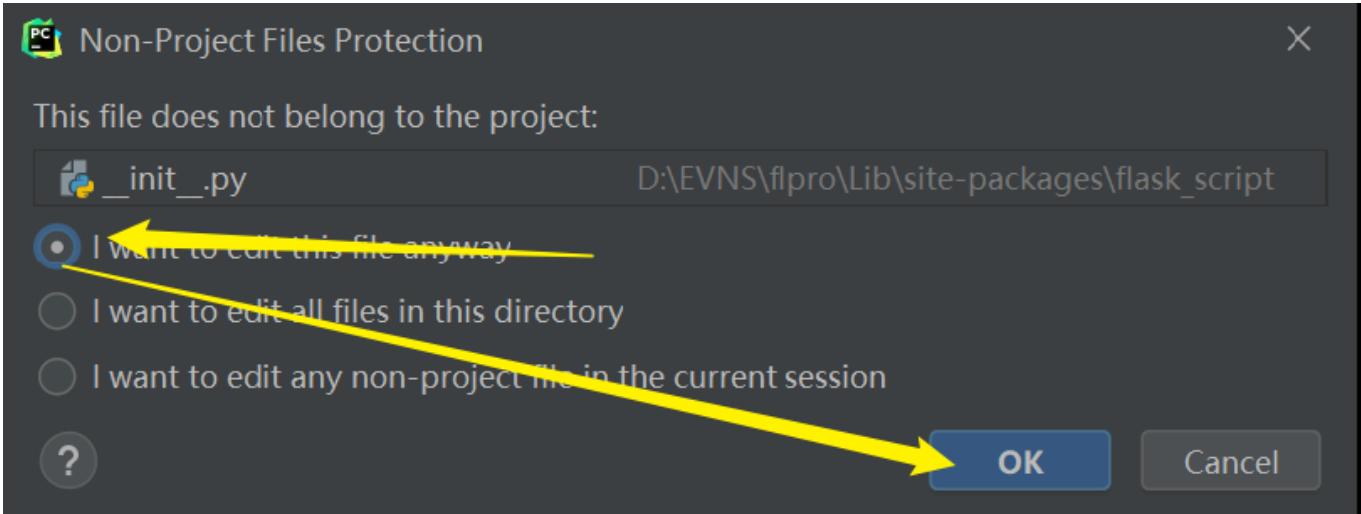
from ._compat import iteritems
from .commands import Group, Option, Command, Server, Shell
from .cli import prompt, prompt_pass, prompt_bool, prompt_choices

__all__ = ["Command", "Shell", "Server", "Manager", "Group", "Option",
           "prompt", "prompt_pass", "prompt_bool", "prompt_choices"]

```

A yellow box highlights the line `# from flask._compat import text_type`, and a yellow arrow points from the terminal output above to this line. The PyCharm interface includes a project tree on the left, toolbars at the top, and various status indicators at the bottom.

操作时出现弹窗，选择如下



修改完成后记得 `ctrl s` 保存修改

再次运行manager文件，信息如下：

File Edit View Navigate Code Refactor Run Tools VCS Window Help flpro - manager.py

flpro manager.py

Project flpro E:\pythonfile\flpro

application.py x manager.py x UrlManager.py x

Run: manager

D:\EVNS\flpro\Scripts\python.exe E:/pythonfile/flpro/manager.py

Map([<Rule '/statics/<filename>' (HEAD, OPTIONS, GET) -> static>])

usage: manager.py [-?] {runserver,shell} ...

positional arguments:

{runserver,shell}

 runserver Runs the Flask development server i.e. app.run()

 shell Runs a Python shell inside Flask application context.

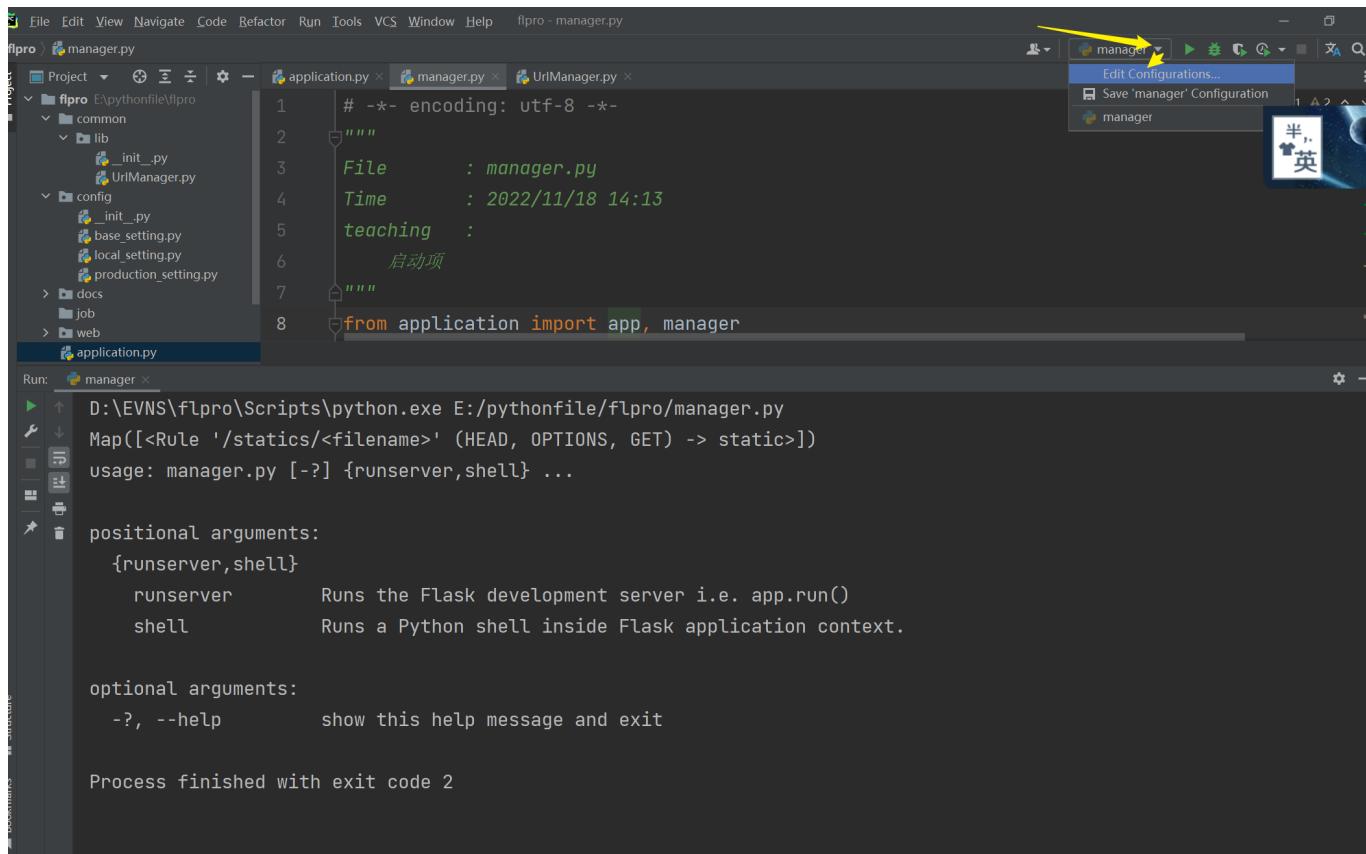
optional arguments:

 -?, --help show this help message and exit

Process finished with exit code 2

此时说明相关配置无问题

仅需进行如下运行配置编辑



File Edit View Navigate Code Refactor Run Tools VCS Window Help flpro - manager.py

flpro > manager.py

Project flpro E:\pythonfile\flpro

 common

 lib

 __init__.py

 UrlManager.py

 config

 __init__.py

 base_setting.py

 local_setting.py

 production_setting.py

docs

job

web

application.py

Run: manager

```
# -*- encoding: utf-8 -*-
"""
File      : manager.py
Time     : 2022/11/18 14:13
teaching  :
启动项
"""

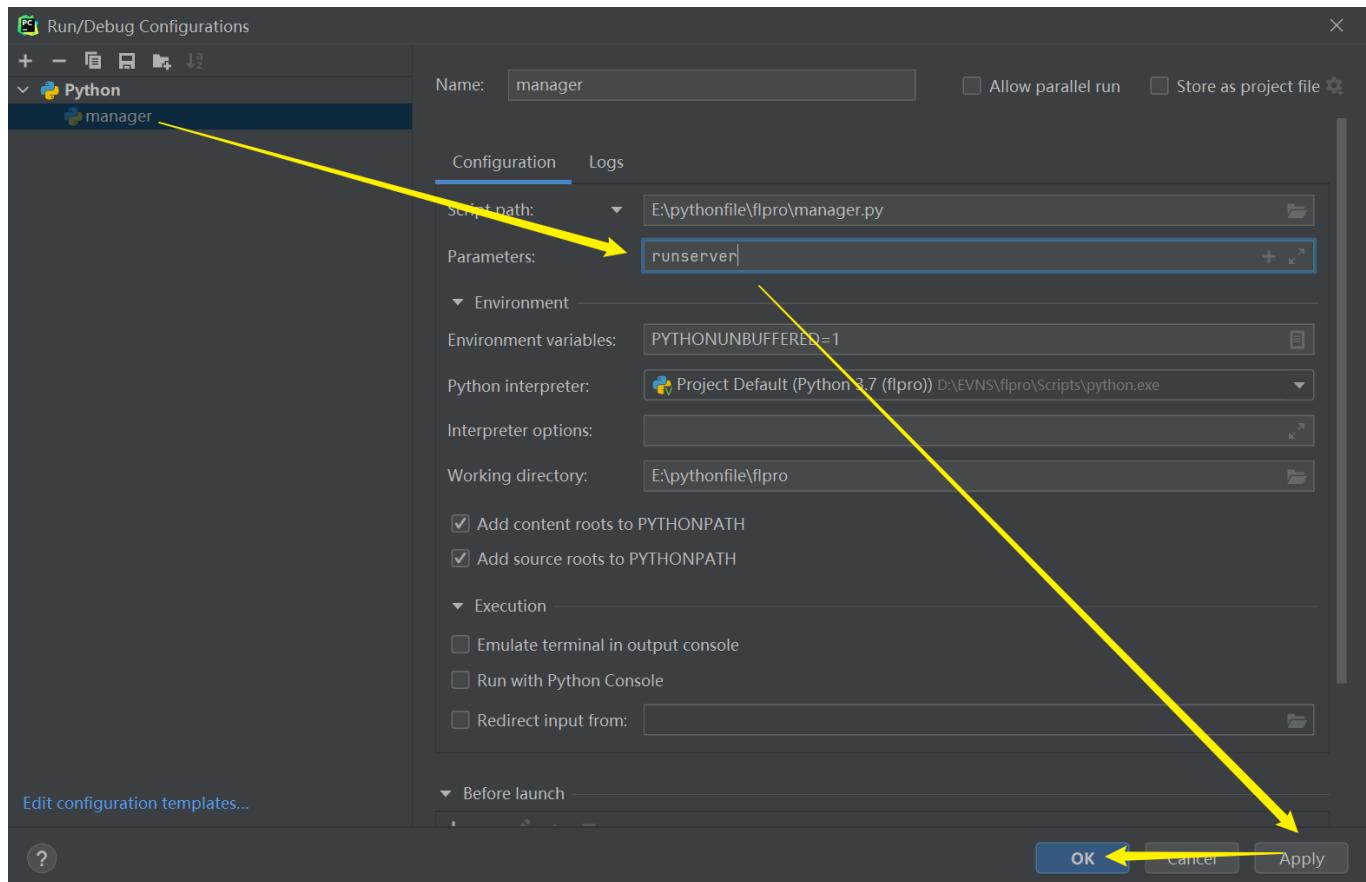
from application import app, manager
```

D:\EVNS\flpro\Scripts\python.exe E:/pythonfile/flpro/manager.py
Map([<Rule '/statics/<filename>' (HEAD, OPTIONS, GET) -> static>])
usage: manager.py [-?] {runserver,shell} ...

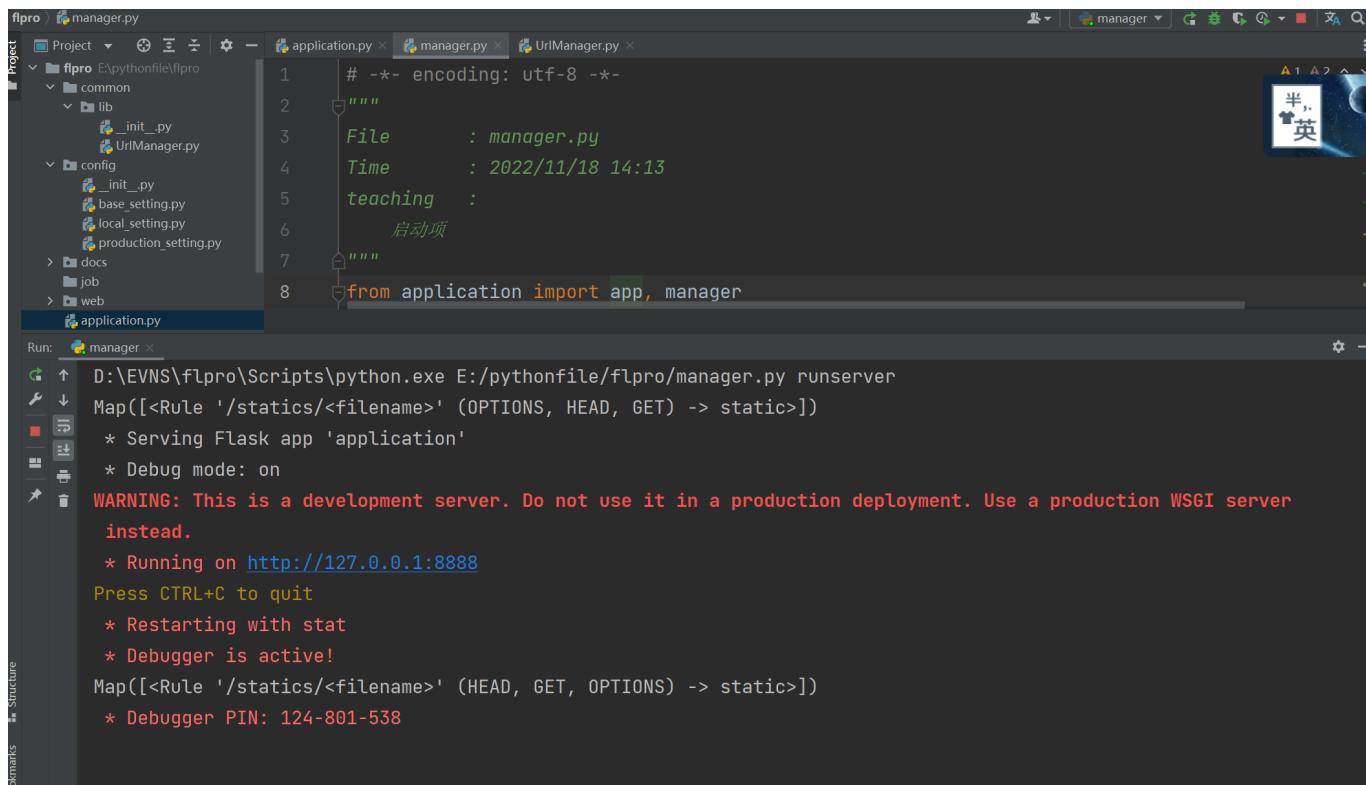
positional arguments:
 {runserver,shell}
 runserver Runs the Flask development server i.e. app.run()
 shell Runs a Python shell inside Flask application context.

optional arguments:
 -?, --help show this help message and exit

Process finished with exit code 2



再次运行就ok了



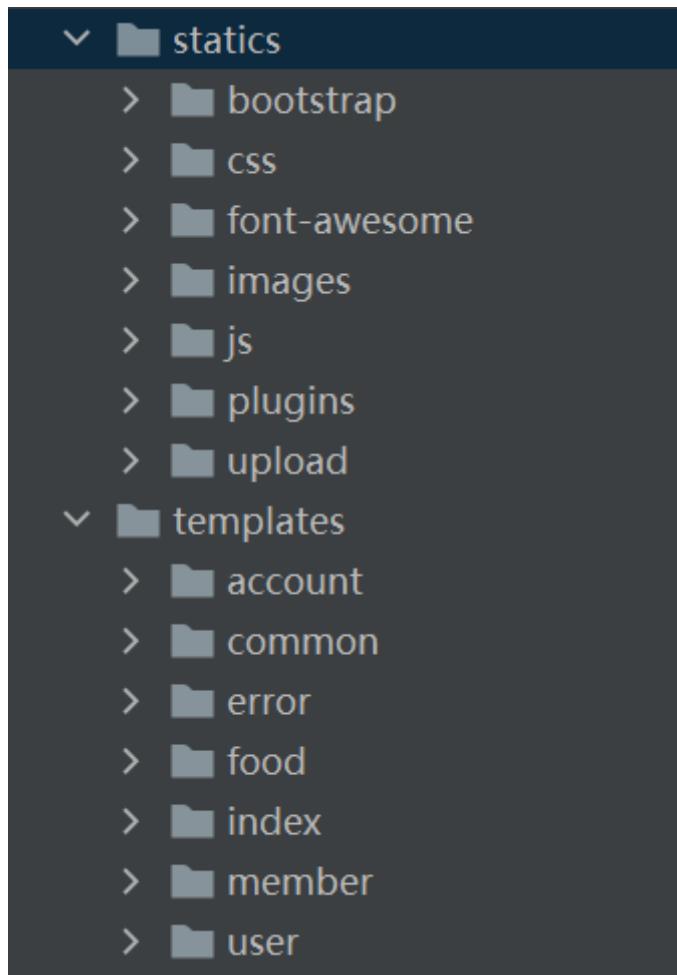
```
# -*- encoding: utf-8 -*-
"""
File      : manager.py
Time     : 2022/11/18 14:13
teaching  :
启动项
"""

from application import app, manager

Run: manager
D:\EVNS\flpro\Scripts\python.exe E:/pythonfile/flpro/manager.py runserver
Map([<Rule '/statics/<filename>' (OPTIONS, HEAD, GET) -> static])
* Serving Flask app 'application'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:8888
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
Map([<Rule '/statics/<filename>' (HEAD, GET, OPTIONS) -> static])
* Debugger PIN: 124-801-538
```

7.项目搭建-引入模板资源

模板资源直接那我项目中的



8.项目搭建-测试访问静态资源

调整配置 application.py中实例应用配置时的static_folder资源指向

The screenshot shows the PyCharm IDE interface. On the left is the Project Explorer pane, displaying the file structure of the 'flpro' project. The 'statics' folder under 'web' contains subfolders like bootstrap, css, font-awesome, images, js, plugins, and upload. Inside the 'images/common' folder are files such as 'avatar.png', 'logo.png', 'mini_qrcode.jpg', and 'qrcode.jpg'. The right pane shows the code editor for 'application.py'. The code is as follows:

```
if platform.system().lower() == "windows":  
    self.config.from_pyfile('config/local_setting.py') # 本地配置  
elif platform.system().lower() == "linux":  
    self.config.from_pyfile('config/production_setting.py') # 可在config下建立一个product  
else: # 通用配置  
    self.config.from_pyfile('config/local_setting.py')  
  
# 初始化数据库对象连接配置  
db.init_app(self)  
  
# 拼接模板目录绝对路径 --- 适应项目上线，测试等移植部署  
template_folder = os.getcwd() + "/web/templates/"  
# print('template_folder:', template_folder)  
  
# 实例应用对象  
app = Application(__name__, template_folder=template_folder, root_path=os.getcwd(),  
                  static_folder=os.getcwd() + "/web/statics/")  
  
# 实例应用管理对象  
manager = Manager(app)  
  
# 配置日志  
log.info("启动成功")
```

The line `static_folder=os.getcwd() + "/web/statics/"` is highlighted with a yellow box.

测试访问：如上图中statics中的images/common/logo.png

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window displaying the file structure of the 'flpro' project. The 'statics' folder under 'web' contains subfolders 'bootstrap', 'css', 'font-awesome', 'images', and 'common'. Inside 'common' are files 'avatar.png', 'logo.png', 'mini_qrcode.jpg', and 'qrcode.jpg'. The right side shows the code editor with 'manager.py' open. The code imports 'application', 'app', 'manager', 'flask_script', and 'Server'. It then imports 'www'. A comment indicates it's a web server. The code then defines a 'runserver' command using 'Server'. The run configuration dropdown at the top shows 'manager' selected. The terminal below shows the command 'D:\EVNS\flpro\Scripts\python.exe E:/pythonfile/flpro/manager.py runserver' being run, followed by the Flask application startup logs. A yellow box highlights the static rule in the logs: 'Map([<Rule '/statics/<filename>' (GET, HEAD, OPTIONS) -> static>])'. A note to the right says '注意观测静态资源访问前缀，依据自身实操为准'.

```
from application import app, manager
from flask_script import Server
import www

# web server --- Server 运行Flask开发服务器, 即app.run()

# manager.add_command("runserver", Server(host='0.0.0.0',
#                                         port=app.config['SERVER_PORT'],
#                                         use_debugger=True,
#                                         threaded=True))

if __name__ == '__main__':
    manager.run()

Map([<Rule '/statics/<filename>' (GET, HEAD, OPTIONS) -> static>])
  * Serving Flask app 'application'
  * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:8888
Press CTRL+C to quit
* Restarting with stat
Map([<Rule '/statics/<filename>' (OPTIONS, GET, HEAD) -> static>])
  * Debugger is active!
```

<http://127.0.0.1:8888/statics/images/common/logo.png>

9.项目搭建-蓝图视图

建立测试视图index

The screenshot shows the PyCharm IDE interface. The Project tool window on the left shows the 'flpro' project structure with 'templates' selected. The code editor on the right shows 'index.py'. The code starts with a她注释 '# -*- encoding: utf-8 -*-', imports 'flask Blueprint render_template', defines a 'route_index' Blueprint, and creates a route '@route_index.route("/")' for the index view. The view function 'def index()' returns a template 'index/index.html'. Yellow arrows point from the 'templates' folder in the Project window to the 'Blueprint' import and the template path in the code editor. Another yellow arrow points from the 'views' folder in the Project window to the 'index.py' file in the code editor.

```
# -*- encoding: utf-8 -*-
from flask import Blueprint, render_template

route_index = Blueprint('index_page', __name__)

@route_index.route("/")
def index():
    return render_template("index/index.html")
```

```
# -*- encoding: utf-8 -*-
from flask import Blueprint, render_template

route_index = Blueprint('index_page', __name__)

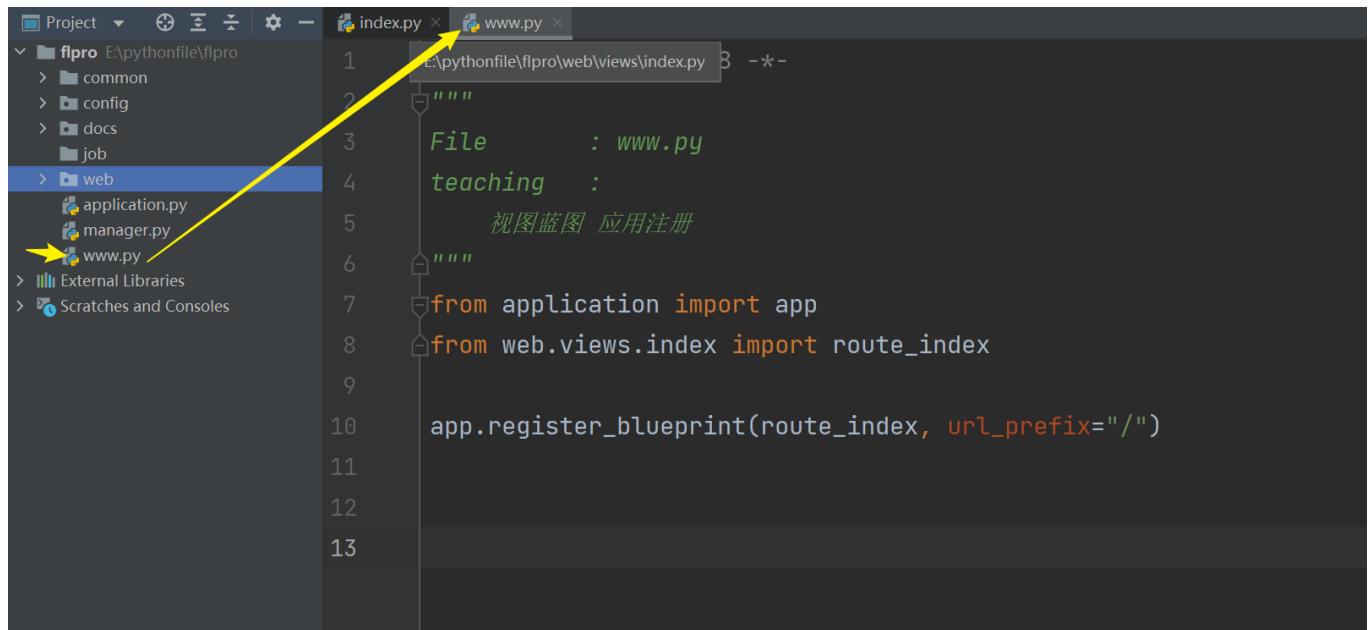
@route_index.route("/")
def index():
    return render_template("index/index.html")
```

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

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



The screenshot shows the PyCharm IDE interface. On the left is the project tree, with 'flpro' selected. Under 'flpro', there are 'common', 'config', 'docs', 'job', and a 'web' folder. Inside 'web', there are 'application.py', 'manager.py', and 'www.py'. The 'www.py' file is currently open in the editor. The code in the editor is:

```
File      : www.py
teaching   :
视图蓝图 应用注册
"""

from application import app
from web.views.index import route_index

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

启动访问测试

<http://127.0.0.1:8888/>

-  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
- 欢迎使用编程浪子订餐管理后台
 -  姓名: xxxx [编辑](#)
 - 手机号码: xxxx
 - [修改密码](#) [退出](#)

日统计

营收概况

1005.00

近30日: 31177.00

静态资源加载失败

经过如下分析 --- 访问前缀static但是我们想默认为statics

```
Map([<Rule '/statics/<filename>' (OPTIONS, GET, HEAD) -> static>,
 <Rule '/' (OPTIONS, GET, HEAD) -> index_page.index>])
* Serving Flask app 'application'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:8888
Press CTRL+C to quit
* Restarting with stat
Map([<Rule '/statics/<filename>' (GET, OPTIONS, HEAD) -> static>,
 <Rule '/' (GET, OPTIONS, HEAD) -> index_page.index>])
* Debugger is active!
* Debugger PIN: 124-801-538
127.0.0.1 - - [18/Nov/2022 18:17:21] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 18:17:21] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 18:17:21] "GET /static/bootstrap/bootstrap.min.css?ver=22222222 HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2022 18:17:21] "GET /static/fontawesome/css/fontawesome.min.css?ver=22222222 HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2022 18:17:21] "GET /static/css/style.css?ver=22222222 HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2022 18:17:21] "GET /static/plugins/jquery-2.1.1.js?ver=22222222 HTTP/1.1" 404 -
```

解决方案: application.py 调整如下--指明前缀为static

```
Project -> fipro -> application.py
application.py
def __init__(self, import_name, template_folder=None, root_path=None, static_fo
    """
    :param import_name:
    :param template_folder: 模板文件
    :param root_path: 应用程序文件的根路径
    """
    # 继承原有初始化
    super(Application, self).__init__(import_name,
                                       template_folder=template_folder,
                                       static_folder=static_folder,
                                       static_url_path='/static/',
                                       root_path=root_path)

    # 加载Flask配置
    self.config.from_pyfile('config/base_setting.py') # 不管本地还是线上都需要的配置

    # 根据系统选择配置加载
    if platform.system().lower() == "windows":
        self.config.from_pyfile('config/local_setting.py') # 本地配置
    elif platform.system().lower() == "linux":
        self.config.from_pyfile('config/production_setting.py') # 可在config下建立一个prod
```

启动访问测试

<http://127.0.0.1:8888/>

管理后台

127.0.0.1:8888

移动设备书签

编程浪子

不掉发的
羊驼

仪表盘

账号管理

美食管理

会员列表

财务管理

统计管理

欢迎使用编程浪子订餐管理后台

营收概况

1005.00

近30日: 31177.00

订单

988

近30日: 29383

会员

358

今日新增: 77 近30日新增: 2454

分享

1250

近30日: 45980

使用highchart画图