# Flask 基础

# 1. 框架介绍

- 重量级的框架：包含**全家桶式**丰富的工具，方便业务程序的**快速开发**，如Django
- 轻量级的框架：只提供Web框架的**核心功能**，自由灵活、**方便高度定制**，如Flask

```
pip install flask==1.1.2
```

# 2. 基本使用

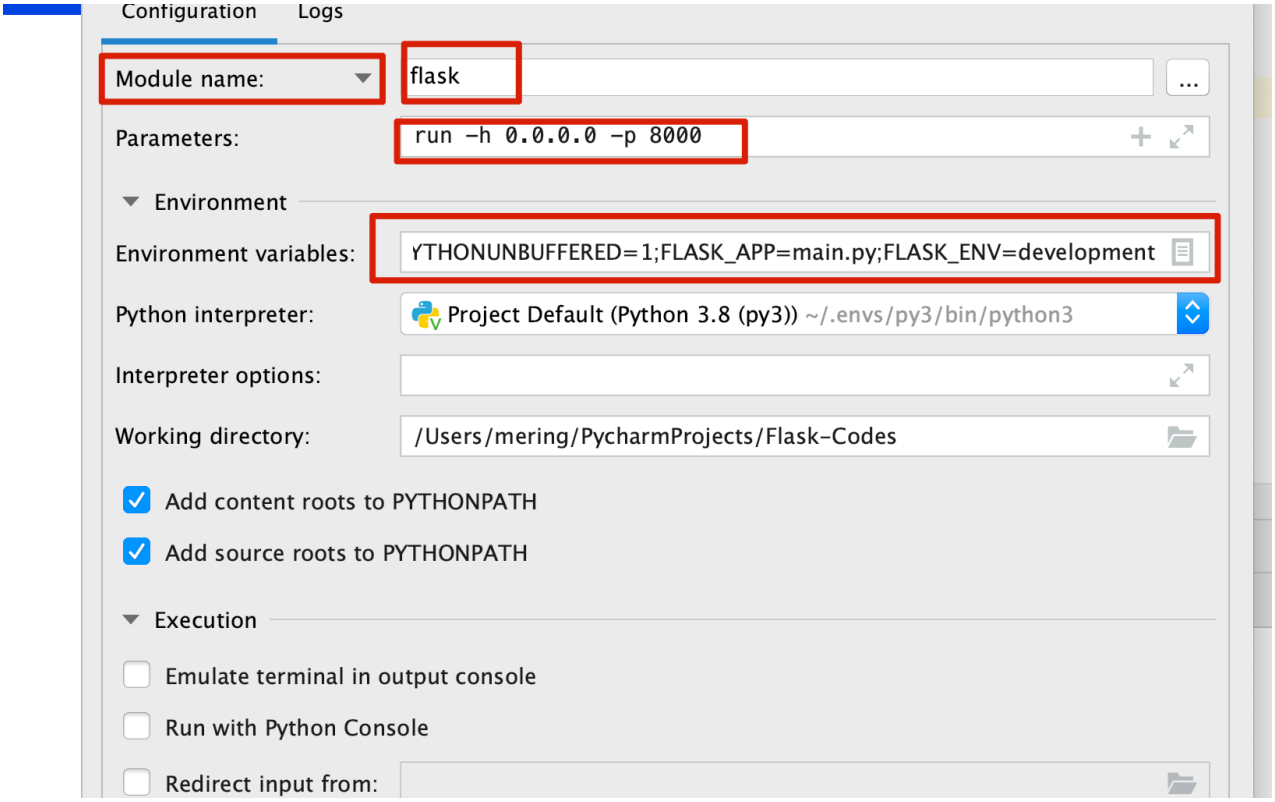
## 最小应用

```python
# TODO:
# 1. 导入Flask类  from flask import Flask
from flask import Flask

# 2. 创建Flask应用对象 固定格式 app = Flask(__name__)
app = Flask(__name__)


# 3. 绑定路由
# @app.route(路径)
# def function_name():
#     return 响应数据

@app.route('/index')
def root():
    # ..... 请求的处理
    # 返回响应
    return 'hello flask'  # 字符串会变成响应体


# 4. 启动web服务 app.run()

app.run()
```

## 命令行运行

```
export FLASK_APP=xx.py  # 指定flask应用所在的文件路径
export FLASK_ENV=development  # 设置项目的环境，默认是生产环境
flask run -h 0.0.0.0 -p 8000  # 启动测试服务器并接受请求
```

## pycharm 运行




# 3. 路由细节

## 请求方法

```
# 3. 绑定路由
# @app.route(路径)
# 细节1 路径以 / 开头
# 细节2 methods=请求方法列表，限制客户端发起的请求方式
# def function_name():
#     return 响应数据
```

如果客户端的请求方法不在列表中，服务端抛出 405 方法不支持错误

```
@app.route('/root', methods=['post', 'get'])
def root():
    return 'root flask'
```

## 路由变量

```python
# 3. 路由变量
# @app.route('/some/path/<变量名>')
# def fun(变量名参数)

@app.route('/user/<user_id>')
def get_user_id(user_id):
    print(user_id)
    return 'get user id'
```

## 路由转换器

```python
# 3. 路由转换器
#   <转化器名(可选的参数):路由变量名>

# string 转化器
# int 转化器

@app.route('/user/<string(minlength=8,maxlength=16):username>')
def get_username(username):
    print(username)
    return 'get username'


@app.route('/userage/<int(min=0,max=200):age>')
def get_userage(age):
    print(age)
    print(type(age))
    return 'get userage'
```

## 自定义路由转换器

```python
# 使用自定义转换器的步骤：
# 1. 定义转换器类，继承 werkzeug.routing.BaseConverter
#     1. 设置 regex 属性（正则匹配规则），用于对路径参数进行匹配，如果请求的路径没有匹配上就返回 404
#     2. 定义 to_python 方法对路径原始参数进行转换，它的返回值就是路由函数参数接受的值
#    这个方法接收一个参数，这个参数就是路径中提取的字符串
# 2. 应用添加自定义转换器 `app.url_map.converters[转换器名]=转换器类`
# 3. 使用自定义路由转换器 @app.route('/some_path/<转换器名:is_vip>')

class MyBoolConverter(BaseConverter):
    regex = '[0NnFfTtYy1]'  # /somepath/<is_vip>/ is_vip [0NnFfTtYy1]

    def to_python(self, value):
        # value 是路径原始字符串
        # 如果 value 在 [TtYy1] 返回 True，否则返回 False
        if value in 'TtYy1':
            return True
        else:
            return False

app.url_map.converters['bool'] = MyBoolConverter


@app.route('/user/<bool:is_vip>')
def user_is_vip(is_vip):
    print(is_vip)
    print(type(is_vip))
    return 'user is vip'
```

如果不满足 regex 匹配，直接返回 404

# 4. 请求数据

## 查询字符串和表单信息

**from flask import request**

```python
@app.route('/request/args_and_form', methods=['post', 'get'])
def args_and_form():
    args = request.args  # 类似于字典
    print(args.get('a', '没有 a 参数'))  # /somepath?a=1&a=2
    # 获取参数所有值 args.getlist(参数名) 返回数组
    print(args.getlist('c'))

    form = request.form  # 类似与字典
    print(form.get('b', '没有 b 参数'))  # age=1 ; age=2
    print(form.getlist('d'))

    return 'read args and form'
args_and_form()
```

如果没有参数，返回空数组

## 请求体和 json 文件

```python
# 第一个路由 读取 data, json
@app.route('/request_body', methods=['post', 'get'])
def read_body():
    print(request.data)  # 原始字节码数据

    print(request.json)  # json 是字典数据

    return 'read body'


# 第二个路由 读取文件, 通过 save 保存到本地, 通过 read 读取原始二进制内容
@app.route('/request_files', methods=['post'])
def read_files():
    files = request.files  # 类似与字典 key: value value 类型 FileStorage

    file1 = files.get('file_1')
    # file.save(路径) 保存到本地文件中
    file1.save('test.png')

    file2 = files.get('file_2')
    # file.read() 返回字节码数据
    bytes_data = file2.read()
    print(bytes_data)

    return 'read files'
```

要求客户端必须上传合法的 json 数据

# 5. 响应

## 静态文件服务

## 返回三个值

```python
# 三元组 响应体, 响应状态, 响应头字典
@app.route('/response_tuple')
def response_tuple():
    return '三元组响应体', 201, {'my-header': 'my-value', 'my-header-2': 'my-value-2'}
```

## 返回响应对象

```python
# 1. resp = make_response(响应体字符串) 返回响应对象
# 2. resp.headers[key]=value
# 3. return resp

@app.route('/response_obj')
def response_obj():
    resp = make_response('自定义响应对象')
    resp.headers['Header-1'] = 'Value-1'

    return resp
```

## 返回 json 重定向

```python
# 第一个路由返回字典--> 相当于返回 json 响应
# 第二个路由重定向到第一个路由

@app.route('/response_json')
def response_json():
    data = {
        'username': 'zhangsan',
        'age': 100
    }
    # django json.dumps(data)
    # flask return data 字典
    return data


@app.route('/response_redirect')
def response_redirect():
    # 重定向 return redirect(重定向的 url)
    return redirect('/response_json')
```

# 6. 状态保持

## cookie

```python
# 定义一个接口，用于读取 cookies 字典数据 request.cookies
@app.route('/cookies')
def read_cookies():
    cookies = request.cookies  # 类似于字典
    print(cookies)
    return 'read cookies'
```

```python
# 定义一个接口，用于颁发 cookie 数据
# 1. 实例化 response 对象 通过 make_response 函数
# 2. response.set_cookie(key,value) 设置 cookie
# 3. 返回 response 对象

@app.route('/cookies/set')
def set_cookies():
    resp = make_response('颁发cookie')
    resp.set_cookie('my-cookie-1', 'my-value-1')
    resp.set_cookie('my-cookie-2', 'my-value-2')

    return resp


# 定义一个接口，用于删除 cookie 数据
# 1. 实例化 response 对象 通过 make_response 函数
# 2. response.delete_cookie(key) 删除 cookie
# 3. 返回 response 对象
@app.route('/cookies/delete')
def delete_cookies():
    resp = make_response('删除cookie')
    resp.delete_cookie('my-cookie-1')
    return resp
```

## session

```python

# 开启 app 的 session
# app.secret_key = 秘钥
app.secret_key = 'testasdfasdf'

# session过期时间 默认31天
print(app.permanent_session_lifetime)  # 打印过期时间
# 设置 session 的过期时间 app.permanent_session_lifetime=timedelta(...)
app.permanent_session_lifetime = timedelta(days=7)

```

# 7. 异常处理

```python
# TODO
# 定义 http 错误处理
# @app.errorhandler(http错误状态码, 400~600)
# def 函数(error): error 表示错误信息对象
#     返回响应
#

@app.errorhandler(404)
def handler_404(error):
    print(error)
    return '你所访问的页面不存在'


# 定义特定异常类错误处理
# @app.errorhandler(异常类)
# def 函数(error): error 表示异常类对象
#     返回响应

@app.errorhandler(ZeroDivisionError)
def handler_zero_div(error):
    print(error)
    return '数据不能除以0'
```

```python
# 定义两个接口，一个接口抛出 http 错误，一个抛出特定异常类对象
#     abort(http错误状态码) 可以抛出 http 错误

@app.route('/abort_404')
def abort_404():
    abort(404)
```

# 0. 练习题

## 定义一个接口(单个知识点)

需求:

- 接收一个路由变量，变量名为 mobile，要求 mobile 是合法的手机号

提示:

- 自定义路由转化器
- `@app.route('/somepath/<自定义路由转换器名:mobile>')`

# 定义一个接口(单个知识点)

需求:

- 接受用户上传的文件，并将文件存储到当前目录下的 static 目录

# 定义一个接口(综合多个知识点)

需求:

- 接受用户提交的 json 请求体
- 如果用户 json 请求体的 username 字段为 admin，password 为 admin1234
    1. 在 session 中保存 is_super_user 为 True
    2. 颁发一个 username cookie 给客户端

提示:

- 读取 json 请求体
- 开启 session 配置
- 使用响应对象颁发 cookie