# 课程介绍

* 1. flask简介;（了解）
* 2. HelloFlask;（掌握）
* 3. Flask的路由系统；（掌握）
* 3. Flask的请求和响应;（掌握）

1. Flask简介
   1. Flask的现状

flask的诞生于2010年的愚人节，本来它只是作者无意间写的一个小玩具，没想到它却悄悄流行起来了。漫长的8年时间，flask一直没有发布一个严肃的正式版本，但是却不能阻挡它成了github上最受好评的Python Web框架。

在2018年4月，flask发布了第一个里程碑的版本1.0版本。

flask内核内置了两个最重要的组件，所有其它的组件都是通过易扩展的插件系统集成进来的。这两个内置的组件分别是werkzeug和jinja2。

* werkzeug是一个用于编写Python WSGI程序的工具包；
* jinja2是一个功能极为强大的模板系统.

Flask的作者是一位澳大利亚的帅哥Armin Ronacher：



* 1. Flask的特点
* 微框架、简洁、只做他需要做的，给开发者提供了很大的扩展性。
* Flask和相应的插件写得很好，用起来很爽。
* 开发效率非常高，比如使用SQLAlchemy的ORM操作数据库可以节省开发者大量书写sql的时间。
* Flask的灵活度非常之高，他不会帮你做太多的决策，一些你都可以按照自己的意愿进行更改，比如：

使用Flask开发数据库的时候，具体是使用SQLAlchemy还是MongoEngine，选择权完全掌握在你自己的手中。

区别于Django，Django内置了非常完善和丰富的功能，并且如果你想替换成你自己想要的，要么不支持，要么非常麻烦。

* 把默认的Jinija2模板引擎替换成其他模板引擎都是非常容易的。
  1. 安装flask

由于Flask的一大特点就是丰富的插件系统，那么每个网站项目，所需要的环境往往都是个性十足，所以建议在做flask开发时，都使用虚拟环境和对应的网站项目适配。

* + 1. 创建虚拟环境

使用virtualenv工具，创建一个flask的工作环境，并激活该环境。

建议使用绝对路径访问所有命令。

D: \Python36\Scripts\virtualenv -p D:\Python36\python.exe D:\virtual\nflask10

激活环境

D:\virtual\nflask10\Scripts\activate

* + 1. 使用pip安装指定flask版本

(nflask10) C:\Users\whn> pip install flask==1.0

安装后，发现目前flask已经内置了一些标准插件了：



1. HelloFlask
   1. 微框架下的helloworld

Flask框架下开发web服务，非常轻便，一个文本编辑器(sublime,vscode,notepad++)编写几行代码，就可以运行自己的网站啦。麻雀虽小，但五脏俱全。

* + 1. 在项目工作目录下编写demo1.py文件

python解释环境文件夹和网站代码文件夹一般建议分开保存，上一节建立了虚拟环境目录，下面建立网站代码的项目空间目录。

mkdir D:\whn\flask-tutorial\demo1

在该目录下，创建一个demo\_site.py文件，编写如下代码：

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/hello/')

def hello\_index():

return "hello world"

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

* + 1. 运行该文件

(nflask10) C:\Users\rocky> python D:\whn\flask-tutorial\demo1\demo\_site.py

* + 1. 测试网站服务

只需要一个python文件，解释器解释后，就可以运行一个简单的web网站了，在浏览器访问：

<http://127.0.0.1:5000/hello/>

就可以看到“hello world”了。

* 1. 代码分析

# 从flask框架中导入Flask类

**from flask import Flask**

# 传入\_\_name\_\_初始化一个Flask实例

**app = Flask(\_\_name\_\_)**

# app.route装饰器映射URL到要执行的函数。这个设置将/hello/的URL映射到了hello\_world函数上。

**@app.route('/hello/')**

def hello\_index():

return 'Hello World!'

**if \_\_name\_\_ == '\_\_main\_\_':**

# 运行本项目，host=0.0.0.0可以让其他电脑也能访问到该网站，port指定访问的端口。

# 默认的host是127.0.0.1，port为5000

**app.run(host='0.0.0.0',port=9000)**

* + 1. \_\_name\_\_ == \_\_main\_\_的意义

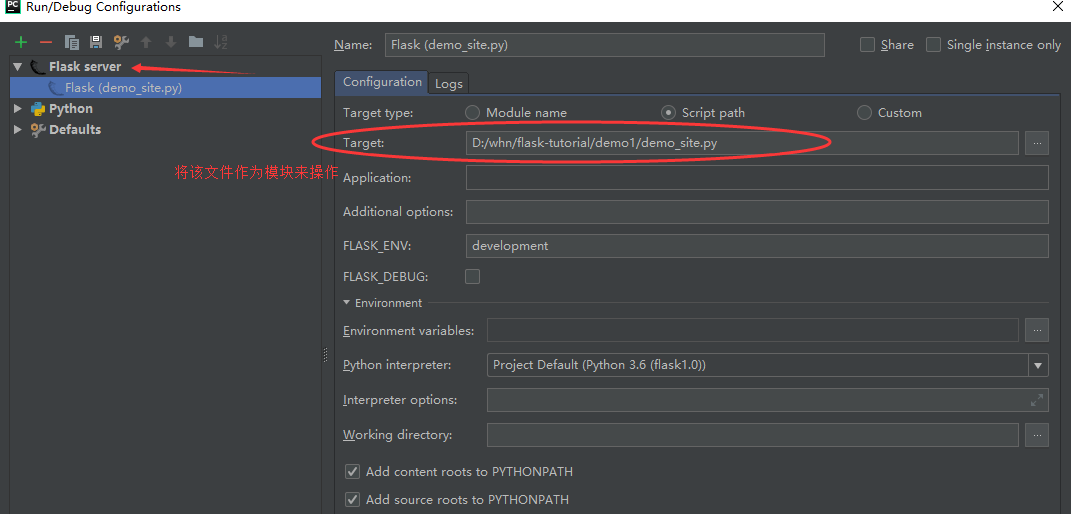
app.run这种方式只适合于开发，他实际还是模拟了一个web服务器来调用python的WSGI接口，所以在实际代码中，这个文件一定不是主模块运行，那么如果没有if \_\_name\_\_这句话，那么当WSGI服务器导入模块后，将会运行2个服务器了。

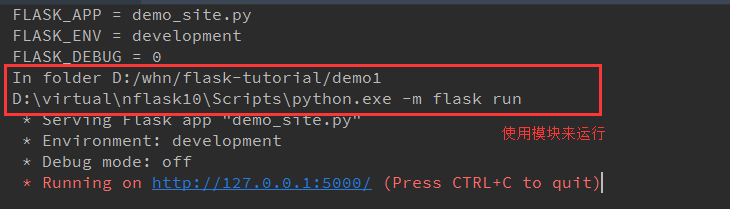
* + 1. 开发时服务器的参数配置
* debug 是否开启调试模式，开启后修改过python代码会自动重启
* threaded 是否开启多线程
* port 启动指定服务器的端口号
* host 主机，默认是127.0.0.1，指定为0.0.0.0代表本机所有ip
  1. Pycharm下的使用

在pycharm下还是提供了两种方式来操作，一种是pycharm自带的创建flask模板的接口，另外一种就是打开一个目录，然后设置工程的解释器即可。

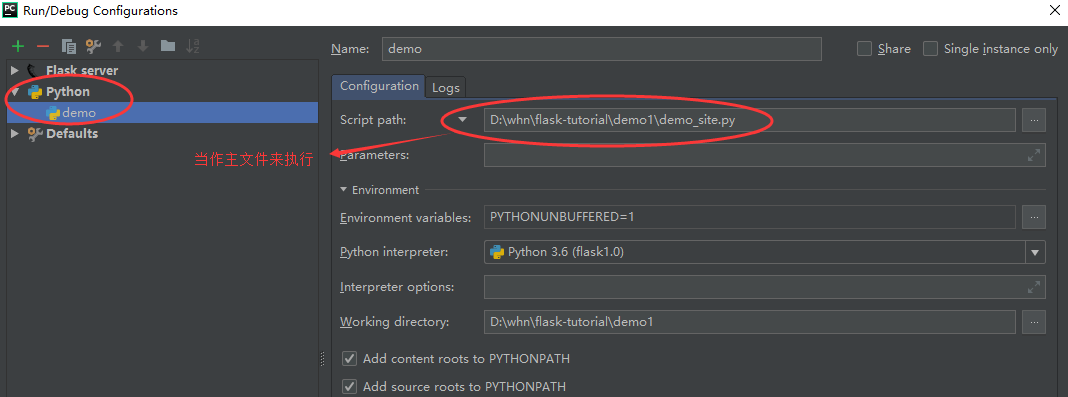
一旦使用pycharm，就要注意pycharm提供的flask运行环境和python程序运行环境是2个不同的配置方式，建议使用python程序运行环境，而不要用pycharm自带的flask调试环境，这样能更好的理解程序运行的机理。

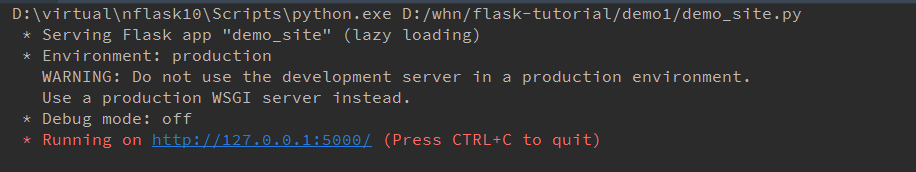
* + 1. pycharm下的flask运行环境配置





* + 1. pycharm下python运行环境配置





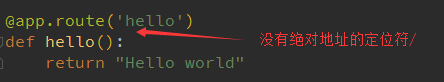
1. Flask下的路由

flask采用了一种优雅的路由注册方式，使用装饰器来把普通函数升级为一个特定URL的视图函数。

下面我们看下Flask中路由的几种表现形式：

* 1. URL表现形式
     1. 直接表示法

第一种是最简单的，直接就跟URL的字符串即可，不过在Flask的路由描述中，必须写成/为开头的字符串，代表这个绝对的资源定位地址信息。



那么在运行时，会报错：



* + 1. url截取法

实际开发时，经常会把一些资源的重要信息，在URL中嵌入进去，例如代表一个店铺的具体商品细节时：

http://127.0.0.1:8000/shop/goods/121/，其中121可能是商品的ID号。那么在Flask里，他提供了一种便捷方法，就是利用<var>的形式捕获，并把捕获的结果以var名字进行视图函数的命名参数传递。

Flask中的这种写法的语法要求是：

**<converter:variable\_name>**

converter代表转换类型：

string 接收任何没有斜杠（‘/’）的文件（默认）

int 接收整型

float 接收浮点型

path 接收路径，可接收斜线（’/’）

uuid 只接受uuid字符串，唯一码，一种生成规则

any 可以同时指定多种路径，进行限定

Flask会自动转换结果，不需要在视图函数里进行转换了。

* + 1. 例子

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return '主页'

@app.route('/login/')

def login():

return 'login'

@app.route('/user/<username>/')

def profile(username):

return '{}的个人中心'.format(username)

* 1. URL产生方式

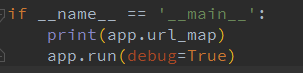
实际开发中，往往需要在a标签，重定向中传递URL信息，一般不会把URL信息以常量形式给出，而是根据一个标签由程序动态获取URL字符串信息。

* + 1. Flask中的路由信息表

在@app.route方法中，我们只注册了URL的信息，并没有指明该URL对应的标签，那该如何构建URL字符串呢？

首先我们先看下在Flask中，他保存的路由映射表的信息。

在Flask中，提供了一个app.url\_map的属性，他里面保存了该app里注册的所有路由映射信息关系表。



保存了当前网站的路由映射表



其中Rule ‘/hello/’，相当于路由匹配的规则，后面的(HEAD, GET, OPTIONS)代表的是该路由可以请求的HTTP的方法。

-> 代表的就是该路由的标签，就是靠他来构建URL字符串的。

默认情况下，Flask会把视图函数的名称当作该URL的标签。

* + 1. URL产生方法

在Flask中，提供了一个url\_for的方法，他提供了通过标签产生URL字符串的方法。

with app.test\_request\_context():

print(url\_for('index'))

print(url\_for('login'))

print(url\_for('login', next='/'))

print(url\_for('profile', username='whn'))

/

/login/

/login/?next=/

/user/whn/

* + 1. app.route的源码分析（了解）

在flask里，会维护路由，视图，标签三者的关系。

路由 ----> endpoint ----> 函数

* 1. URL请求方法

Flask在注册路由时，可以指定该函数接收哪些HTTP请求的动作（方法）

@app.route(‘/hello/’, methods=[‘GET’,’POST’])

methods中指定请求方法

GET

POST

HEAD

PUT

DELETE

1. Flask下的请求

服务器在接收到客户端的请求后，Flask会自动创建Request对象，Request对象不可修改。

在Flask中，视图函数并没有传递request对象，这个对象将是Flask框架内部维护的一个类全局变量，我们只需要从flask框架中导入这个变量即可。注意这个是小写字母。

from flask import request

属性

url 完整请求地址

base\_url 去掉GET参数的URL

host\_url 只有主机和端口号的URL

path 路由中的路径

method 请求方法

remote\_addr 请求的客户端地址

**args GET请求参数**

**form POST请求参数**

files 文件上传

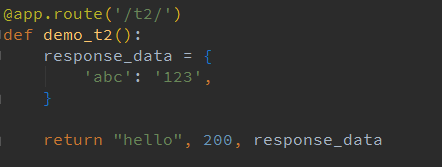
**headers 请求头**

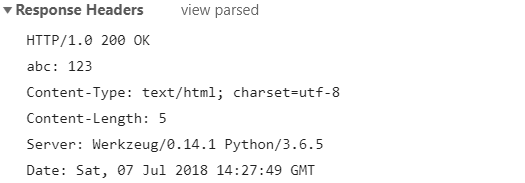
**cookies 请求中的cookie**

可以结合postman工具进行测试，以熟悉这些属性。

1. Flask下的响应

Flask的视图函数的返回值其实是一个元组。





@app.route('/t2/')

def demo\_t2():

response\_data = {

'abc': '223',

}

x = make\_response("h123", 200, response\_data)

x.headers['t2'] = 'bbc'

return x

Flask提供的HTTP响应返回的方式：

* + - 1. make\_response产生的对象；
      2. 字符串单一参数返回后，默认使用200，text/html形式构造响应头；
      3. jsonify，返回json格式的快捷方式；

1. 课程总结
   1. 重点
      * 1. Flask的路由系统
        2. Flask的请求，响应
   2. 难点
      * 1. 和Django的对比
   3. 如何掌握？
      * 1. 通过打印信息看
        2. 利用断点查看信息
2. 课后练习
   * + 1. 将课上内容重新做一遍
3. 面试题
   * + 1. flask和Django的路由区别
4. 扩展知识或课外阅读推荐（可选）
   1. 课外阅读