Flask框架讲解

在我的理解看来，Flask就是结合Python和HTML编写web页面。Python语言我们可以用来编写URL、返回结果；再通过HTML的调用进而形成WEB页面~~是不是说的有点啰嗦？？具体看看大家就清楚了~~

1. Flask是什么

Flask是一个轻量级的可定制框架，使用Python语言编写，较其他同类型框架更为灵活、轻便、安全且容易上手。

Flask框架的主要特征是核心构成比较简单，但具有很强的扩展性和兼容性，程序员可以使用Python语言快速实现一个网站或Web服务。

1. 安装Flask

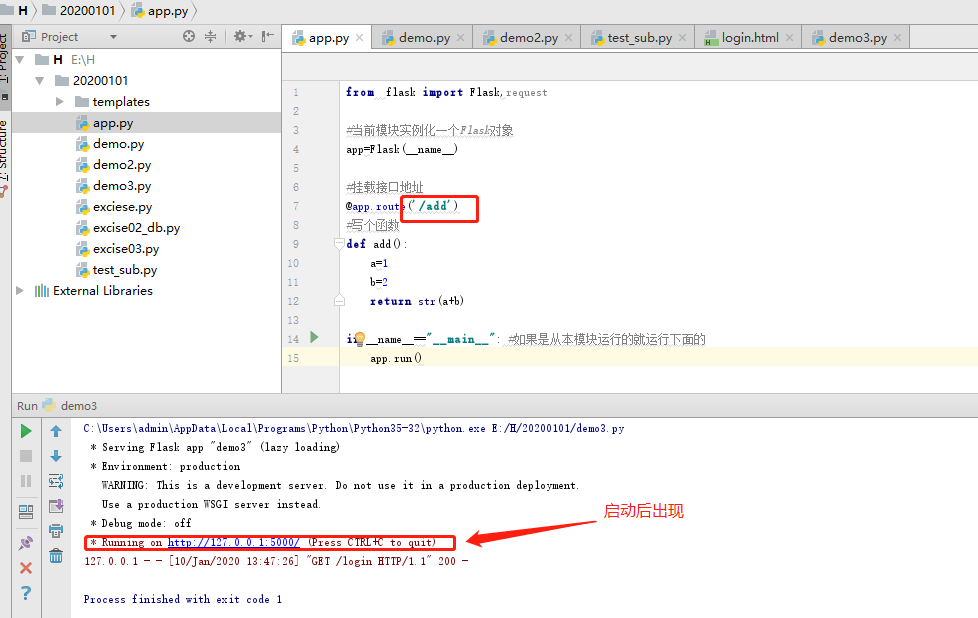
pip install flask

1. 打开Pycharm编辑器，新建一个py文件。如：denmo.py

**from** flask **import** Flask

*#当前模块实例化一个Flask对象*app=Flask(\_\_name\_\_)  
*#挂载接口地址*@app.route(**'/add'**)  
*#写个函数***def** add():  
 a=1  
 b=2  
 **return** a+b  
**if** \_\_name\_\_==**"\_\_main\_\_"**: *#如果是从本模块运行的就运行下面的* app.run()

运行Py文件：



出现上述的结果说明运行成功，现在我们打开网页输入URL：<http://127.0.0.1:5000/add>

就可以看到返回结果了~~

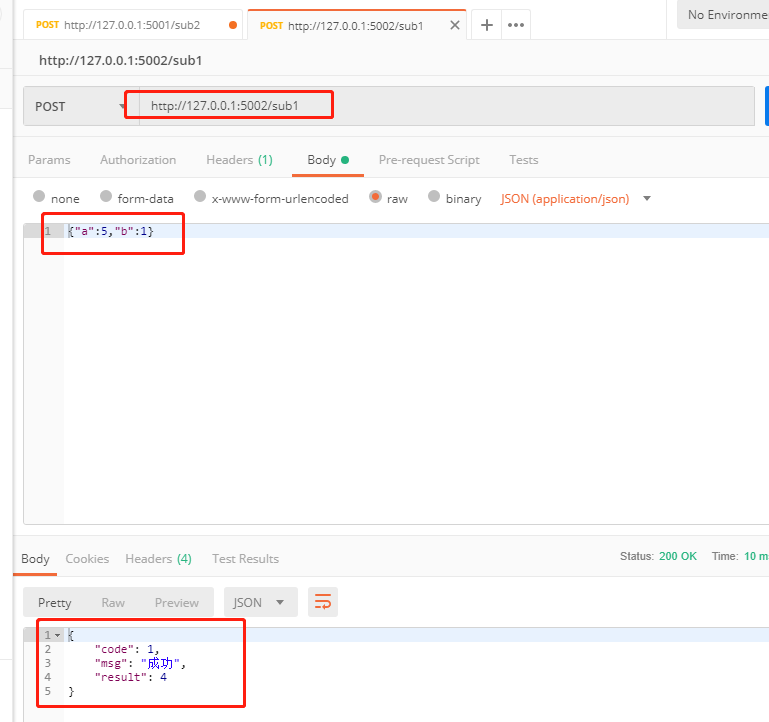
这里 需要注意的地方是：return a+b如果是这种写法的话返回会报错，因为此时返回的是一个INT类型的数据，不支持~~~

好了，上述就是一个比较简单的Flask的写法，下面给大家讲讲如何利用Flask发送POST/Get请求呢？

**from** flask **import** Flask,request,jsonfy

*#当前模块实例化一个Flask对象*app=Flask(\_\_name\_\_)  
@app.route(**'/sub1'**,methods=[**"post"**,**'get'**]) *#接口类型***def** reduce():  
 a = request.json.get(**'a'**)  
 b = request.json.get(**'b'**)  
 final = int(a) - int(b)  
 print(final)  
 **return** jsonify({**"code"**:1,**"msg"**:**"成功"**,**"result"**:str(final)})

如上述写法，jsonfy是Flask下让其返回为json格式；此时我们可以通过POSTMan工具进行访问~~说明接口编写成功！！！！！



拓展：当我们写一个接口时，针对不同的输入结果会有不同的返回值，那Flask框架下怎么具体操作呢？来来来~~

比如上诉写的减法可能会出现的情况：

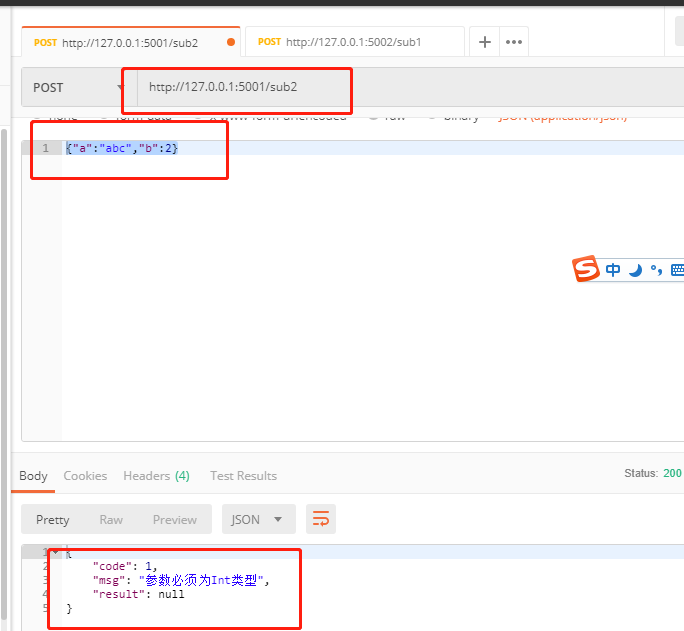
*1.a=39,b=12 27  
2.a=10 ,b 参数缺失  
3.a='abc',b=3 参数必须为int*

等等还有其他的情况我们暂时就看这几种~~~

**from** flask **import** Flask,request,jsonfy

app=Flask(\_\_name\_\_)  
@app.route(**'/sub2'**,methods=[**"post"**]) **def** reduce():  
 a = request.json.get(**'a'**)  
 b = request.json.get(**'b'**)  
 **if** a **is None or** b **is None**:  
 **return** jsonify({**"code"**:0,**"msg"**:**"参数缺失"**,**"result"**:**None**})  
 **if not** isinstance(a,int) **or not** isinstance(b,int):  
 **return** jsonify({**"code"**: 1, **"msg"**: **"参数必须为Int类型"**, **"result"**: **None**})  
  
 final = a-b  
 **return** jsonify({**"code"**:2,**"msg"**:**"成功"**,**"result"**:str(final)})

写完后继续POSTMAN调用查看是否成功：



好了，Flask的接口编写就先介绍到这里，下面给大家讲讲“

1. 新建文件夹：templates[固定写法]
2. 新建login.html

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 *<!-- 新 Bootstrap4 核心 CSS 文件 -->* <**link rel="stylesheet" href="https://cdn.staticfile.org/twitter-bootstrap/4.3.1/css/bootstrap.min.css"**>  
 <**title**>Title</**title**>  
</**head**>  
<**body**>  
 <**h1**>接口测试报告<**h1**>  
 <**h2**>执行人：{{tester}}<**h2**> #2个括号表示变量  
 <**p**>成功：{{pass\_NO}}</**p**>  
 <**p**>失败：{{fail\_NO}}</**p**>  
 <**table class="table"**>  
 <**div class="container"**>  
 <**tr**><**td**>用例名称</**td**><**td**>执行结果</**td**></**tr**>  
  
 {% for item in data %}  
 <**tr**><**td**>{{item['case']}}</**td**><**td**>{{item['result']}}</**td**></**tr**>  
 {% endfor %}  
 </**div**>  
 </**table**>  
</**body**>  
</**html**>

1. 编写demo3.py

**from** flask **import** Flask,request,jsonify,render\_template  
app=Flask(\_\_name\_\_)  
data=[  
 {**'case'**:**'test\_sub01'**,**'result'**:**"PASS"**},  
 {**'case'**: **'test\_sub02'**, **'result'**: **"FAIL"**},  
 {**'case'**: **'test\_sub03'**, **'result'**: **"PASS"**},  
 {**'case'**: **'test\_sub04'**, **'result'**: **"PASS"**}  
]  
@app.route(**'/login'**)  
**def** login():  
 **return** render\_template(**'login.html'**,tester=**"倩倩"**,pass\_NO=4,fail\_NO=1,data=data)  
**if** \_\_name\_\_==**"\_\_main\_\_"**:  
 app.run()