Flask安装部署

Flask是一个轻量级可扩展的由Python开发的WEB框架.

Flask的安装.
(proj_b) [root@backup-platform proj_b]# pip install flask

基本概念:

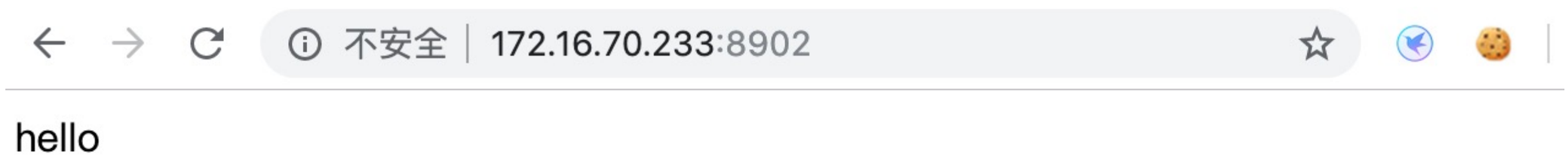
1)程序实例:
所有的Flask项目都必须创建一个程序实例, 代码如下:
from flask import Flask
app = Flask(__name__)

2)路由:
客户端把请求发送给web服务器, web服务器把请求发送给Flask程序实例, 程序实例需要知道每个url请求运行哪些代码, 所以保存了一个URL到Python函数的映射关系; 用来处理URL跟函数之间关系.Flask 通过app.route装饰器把修饰的函数注册成为路由. 示例代码如下:

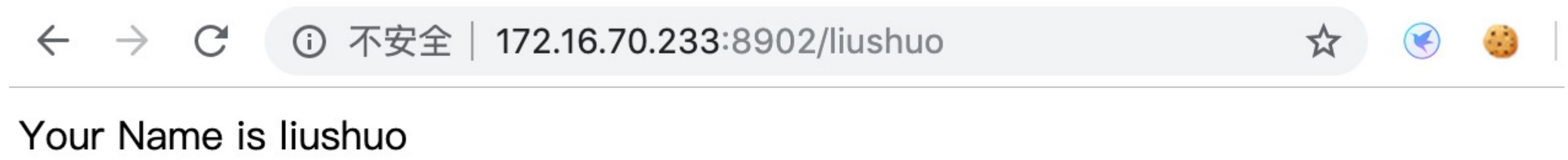
In [ ]:
```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return 1


@app.route('/<name>')
def get_name(name):
    return 'Your Name is %s'%name

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8902)
```

当用户请求的URL为网站的根路径, 则该HTTP请求交给index函数进行处理. 同时Flask也可以处理动态的url如函数get_name.

视图函数1:



视图函数2:



二. 请求-响应循环
请求上下文
Flask中有两种上下文, 程序上下文(application context)和请求上下文(request context)

变量名
current_app:指向处理请求的当前程序实例.

g: 存储在程序上下文中, 而程序上下文会随着每一个请求的进入而激活, 随着每一个请求的处理完毕而销毁, 所以每次请求都会重设这个g值.

request: 当请求进入时, flask会自动激活请求上下文, 这时可以使用request和session变量. 当请求上下文被激活时, 程序上下文也被自动激活.

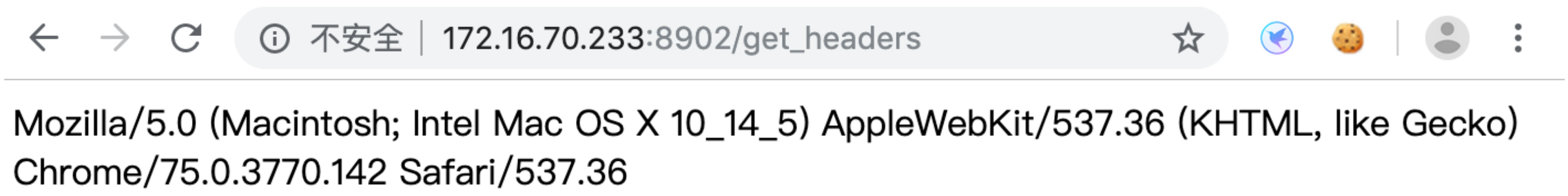
session: 用于存储请求之间需要记住的值的词典, 解决Http协议无状态的问题

Flask从客户端收到请求时, 要让视图函数能访问一些对象. 这样才能处理请求, request对象就是一个很好的例子, 它封装了客户端发送的HTTP请求.

示例代码:

In [ ]:
```
@app.route('/get_headers')
def get_person():
    s = request.headers.get('User-Agent')
    return s
```

视图输出:



请求钩子:
有时在处理请求之前或之后执行代码很有用, 比如在请求开始时可能需要创建数据库连接 或者认证发起请求的用户, 为了避免在视图函数中使用重复的代码, Flask提供了注册通用函数的功能, 注册的函数可在请求被分发到视图函数之前或之后调用.

before_first_request
#注册一个函数在处理第一个请求之前运行
before_request
#注册一个函数在每次请求之前运行

after_request
#注册一个函数，如果没有未处理的异常抛出，在每次请求之后执行.
teardown_request
#注册一个函数，即使有未处理的异常抛出，也在每次请求之后执行.

比如在请求钩子函数和视图函数之间共享数据一般使用上下文全局变量g，例如before_request处理程序可以从数据库加载已登录的用户，并将其保存到 g.user中，视图函数可以通过g.user获取已登录的用户.

响应:
Flask调用视图函数以后，会将其返回值作为响应的内容；响应值作为HTML页面发送回给客户端，响应中的状态码是一个很重要的部分，这个状态码表示请求已经被成功的处理
如果视图函数返回的响应需要不同的状态码，那么可以把数字代码作为第二个返回值 添加到文本之后. 示例代码:

In [ ]:
```
@app.route('/bad')
def bad():
    return 'bad page', 400
```

在Http响应中设置Cookie的方式.

In [ ]:
```
@app.route('/')
def index():
    response = make_response('hello')
    response.set_cookie('ansower', 42)
    return response
```

重定向:
有一种名为重定向的特殊响应类型，这种响应没有页面文档，只告诉浏览器用一个新地址用来加载页面.
示例代码:

In [ ]:
```
from flask import redirect
@app.route('/jump')
def jump():
    return redirect('http://www.baidu.com')
```

还有一种特殊的响应由abort函数生成，用于处理错误. 下面的例子中如果用户的id为0,则返回状态码404; 示例代码:

In [ ]:
```
@app.route('/user/<id>')
def bad(id):
    if id == 0:
        abort(404)
    else:
        return 'user id is %s' %id
```

abort不会吧控制器交还给调用它的函数，而是抛出异常把控制器交给web服务器.