


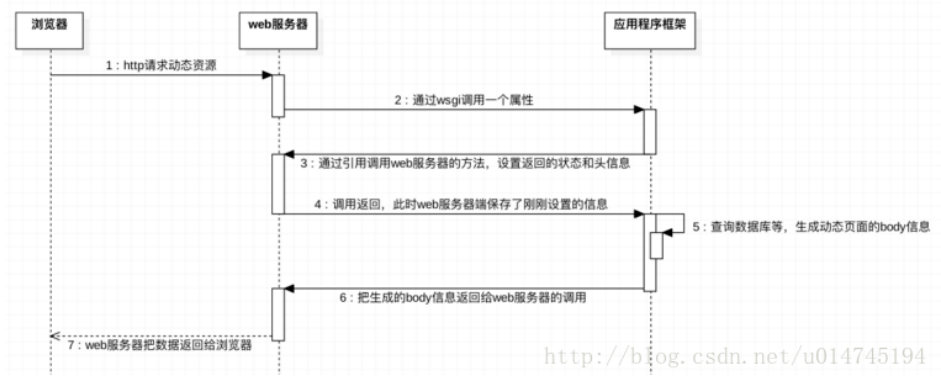
Python中WSGI接口的理解

作者 晓可加油 (/u/90d6604634cb) [+ 关注](#)

2017.05.12 12:05 字数 1249 阅读 54 评论 0 喜欢 0

(/u/90d6604634cb)

Num01-->浏览器动态请求页面流程图



这里写图片描述

以上图片就是整个浏览器动态请求服务器的全过程。

Num02-->什么是WSGI?

WSGI，全称 Web Server Gateway Interface，或者 Python Web Server Gateway Interface，是为 Python 语言定义的 Web 服务器和 Web 应用程序或框架之间的一种简单而通用的接口。

WSGI 的官方定义是，the Python Web Server Gateway Interface。从名字就可以看出来，这东西是一个Gateway，也就是网关。网关的作用就是在协议之间进行转换。

WSGI 是作为 Web 服务器与 Web 应用程序或应用框架之间的一种低级别的接口，以提升可移植 Web 应用开发的共同点。WSGI 是基于现存的 CGI 标准而设计的。

很多框架都自带了 WSGI server，比如 Flask，webpy，Django、CherryPy等等。当然性能都不好，自带的 web server 更多的是测试用途，发布时则使用生产环境的 WSGI server或者是联合 nginx 做 uwsgi。

一句话总结就是：WSGI就像是一座桥梁，一边连着web服务器，另一边连着用户的application应用。但是呢，这个桥的功能很弱，有时候还需要别的桥来帮忙才能进行处理。

Num03-->WSGI的作用

WSGI有两方：“服务器”或“网关”一方，以及“应用程序”或“应用框架”一方。服务方调用应用方，提供环境信息，以及一个回调函数（提供给应用程序用来将消息头传递给服务器方），并接收Web内容作为返回值。

所谓的 WSGI中间件同时实现了API的两方，因此可以在WSGI服务和WSGI应用之间起调解作用：从WSGI服务器的角度来说，中间件扮演应用程序，而从应用程序的角度来说，中间件扮演服务器。

“中间件”组件可以执行以下功能：

- 1，重写环境变量后，根据目标URL，将请求消息路由到不同的应用对象。
- 2，允许在一个进程中同时运行多个应用程序或应用框架。
- 3，负载均衡和远程处理，通过在网络上转发请求和响应消息。
- 4，进行内容后处理，例如应用XSLT样式表。

Num04-->WSGI Application

可调用对象是指：函数、方法、类或者带有callable方法的实例。

可调用对象都可以作为the application object

WSGI协议要求：

the application object接受两个参数且可以被多次调用

这两个参数分别为：

参数一：environ:是一个CGI式的字典；

参数二：start_response:是一个回调函数：application用来向server传递http状态码/消息/http头

注意：协议要求可调用对象必须将响应体封装成一个可迭代的strings返回。

```
# the application object. 可以使用其他名字,
# 但是在使用mod_wsgi 时必须为 "application"
def application( environ, start_response):

# 构造响应体, 以可迭代字符串形式封装
    response_body = 'The request method was %s' % environ['REQUEST_METHOD']

# HTTP 响应码及消息
    status = '200 OK'

# 提供给客户端的响应头.
# 封装成list of tuple pairs 的形式:
# 格式要求: [(Header name, Header value)].
    response_headers = [('Content-Type', 'text/plain'),
                        ('Content-Length', str(len(response_body)))]

# 将响应码/消息及响应头通过传入的start_reponse回调函数返回给server
    start_response(status, response_headers)

# 响应体作为返回值返回
# 注意这里被封装到了list中.
    return [response_body]
```

Num05-->WSGI Server



wsgi server可以理解为一个符合wsgi规范的web server，接收request请求，封装一系列环境变量，按照wsgi规范调用注册的wsgi app，最后将response返回给客户端。

WSGI server必须要调用application，同时，从application的协议要求可知：

第一：WSGI server必须向application提供环境参数，因此，自身也必须能够获取环境参数。

第二：WSGI server接收application的返回值作为响应体。

```
#最简单的WSGI server为Python自带的wsgiref.simple_server
from wsgiref.simple_server import make_server
srv = make_server('localhost', 8080, fuc)
srv.serve_forever()
```

wsgi server的基本工作流程：

- 1，服务器创建socket，监听端口，等待客户端连接。
- 2，当有请求来时，服务器解析客户端信息放到环境变量environ中，并调用绑定的handler来处理请求。
- 3，handler解析这个http请求，将请求信息例如method，path等放到environ中。
- 4，wsgi handler再将一些服务器端信息也放到environ中，最后服务器信息，客户端信息，本次请求信息全部都保存到了环境变量environ中。
- 5，wsgi handler 调用注册的wsgi app，并将environ和回调函数传给wsgi app
- 6，wsgi app 将response header/status/body 回传给wsgi handler
最终handler还是通过socket将response信息塞回给客户端。

Num06-->WSGI MiddleWare

假设一个符合application标准的可调用对象，它接受可调用对象作为参数，返回一个可调用对象的对象。

那么对于server来说，它是一个符合标准的可调用对象，因此是application。

而对于application来说，它可以调用application，因此是server。

这样的可调用对象称为middleware。



```
#一个路由的例子实例如下:
import re

# 这是一个标准的index application object
def index(envIRON, start_response):
    start_response('200 OK', [('Content-Type', 'text/html')])
    return ['index page']

# 这是一个标准的hello application object
def hello(envIRON, start_response):
    start_response('200 OK', [('Content-Type', 'text/html')])
    return ['hello page']

# 这是一个标准的not _found application object
def not_found(envIRON, start_response):
    start_response('404 NOT FOUND', [('Content-Type', 'text/plain')])
    return ['Not Found Page']

# map urls to functions
urls = [
    (r'^$', index),
    (r'hello/?$', hello)
]

# 这是一个middleware
# 根据不同的route返回不同的application object
def application(envIRON, start_response):
    path = envIRON.get('PATH_INFO', '').lstrip('/')
    for regex, callback in urls:
        match = re.search(regex, path)
        if match is not None:
```

本文参考资料如下:
<http://www.jb51.net/article/65875.htm>

📅 日记本 (/nb/7928623)

举报文章 © 著作权归作者所有



晓可加油 (/u/90d6604634cb) ♂

写了 47521 字, 被 20 人关注, 获得了 16 个喜欢

(/u/90d6604634cb)

+ 关注

热爱Python技术, 热爱Android技术, 热爱前端开发, 热爱人工智能研发 毕生信念: 技术改变人生, 技术改...

如果觉得我的文章对您有用, 请随意赞赏。您的支持将鼓励我继续创作!

赞赏支持

❤ 喜欢 1



更多分享

(<http://cwb.assets.jianshu.io/notes/images/12331002/>)



写下你的评论...

评论

