# What’s WSGI

WSGI，是Python Web Server Gateway Interface的简称，是web底层跟application解耦的协议

## 实现的功能：

**服务器层（server）:**

1. 将来自socket的数据包解析为http协议。
2. 调用Framwork的application方法。

**应用服务器层（Framwork/application）:**

1. 实现application方法。
2. 根据不同的路由，调用不同的实现方法

# 实现server

1. 通过socket监听指定端口，从端口中读取数据（使用多进程）
2. 处理http请求头

res\_socket=splitlines() # 切割字符串  
# 处理头信息

path = re.match( # 获取GET /index.html HTTP/1.1

r"\w+\s+(/[^\s]\*)\s+", str\_info[0].decode("utf-8")

).group(1)

# 获取

1. 组成WSGI环境参数

env={“PATH\_INFO”:path}

1. 调用框架--**核心**

# self.\_\_start\_response -> 是一个回调函数

# 这个函数是调用Farmwork中的application方法

# 因为**响应信息**，**响应头**需要Farmwork定义，在server中无法定义，而server还要调用这些

# 数据，所以使用回调函数将Farmwork中定义的数据取回来

body = self.application(env, self.\_\_start\_response)

1. 定义\_\_start\_response方法

# server文件中定义

# 定义响应头，拼接响应头

response\_headers = "HTTP/1.1 " + status + "\r\n"

for header in headers:

response\_headers += "%s: %s\r\n" % header

# 将信息保存到server中的 **实例变量** 中

self.response\_headers = response\_headers

1. 拼接返回头

client.send(bytes(response, "utf-8"))

1. 关闭连接

client.close()

# 实现框架Farmwork

1. 主要实现application方法

def application(environ, start\_response):

start\_response('200 OK', [('Content-Type', 'text/html')])

return '<h1>Hello, web!</h1>'

1. start\_response是server中定义的\_\_start\_response方法，通过这个方法将响应头，状态码返回，这是一个回调方法