如何理解Nginx, WSGI, Flask之间的关系

概览

之前对 Nginx，WSGI（或者 uWSGI，uwsgi），Flask(或者 Django），这几者的关系一存存在疑惑。通过查阅了些资料，总算把它们的关系理清了。

总括来说，客户端从发送一个 HTTP 请求到 Flask(django) 处理请求，分别经过了 web 服务器层，WSGI层，web框架层，这三个层次。不同的层次其作用也不同，下面简要介绍各层的作用。

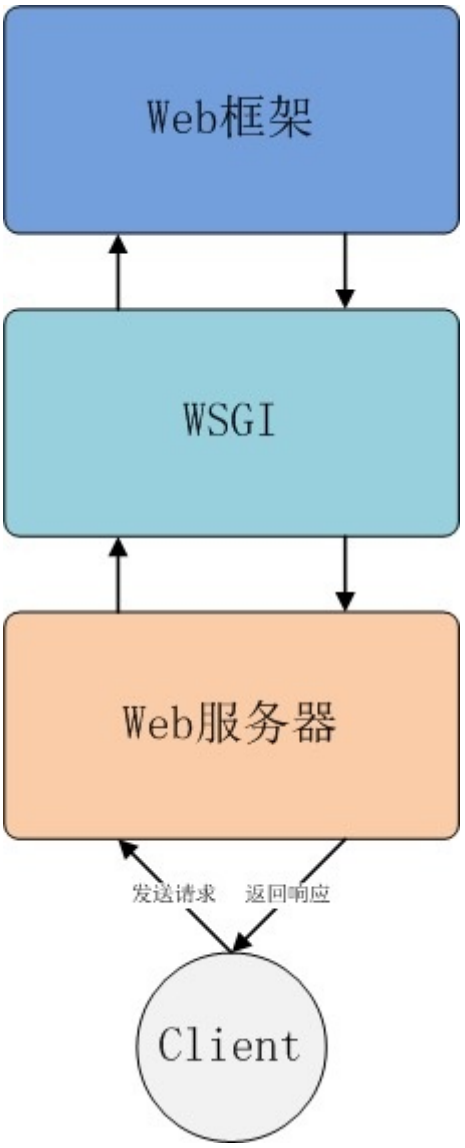

图1：web服务器，web框架与 WSGI 的三层关系

Web服务器层

对于传统的客户端 - 服务器架构，其请求的处理过程是，客户端向服务器发送请求，服务器接收请求并处理请求，然后给客户端返回响应。在这个过程中，服务器的作用是
1.接收请求
2.处理请求
3.返回响应

Web服务器是一类特殊的服务器，其作用是主要是接收 HTTP 请求并返回响应。提起 web 服务器大家都不会陌生，常见的 web服务器有 **Nginx，Apache，IIS**等。在上图1的三层结构中，web服务器是最先接收用户请求的，并将响应结果返回给用户。

Web框架层

Web框架的作用主要是方便我们开发 web应用程序，HTTP请求的动态数据就是由 web框架层来提供的。**常见的 web框架有Flask，Django**等，我们以 Flask 框架为例子，展示 web框架的作用：

```
from flask import Flask
app = Flask(__name__)

@app.route('/hello')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

以上简单的几行代码，就创建了一个 web应用程序对象 app。app 监听机器所有 ip 的 8080 端口，接受用户的请求连接。我们知道，HTTP 协议使用 URL 来定位资源，上面的程序会将路径 /hello 的请求交由 hello_world 方法处理，hello_world 返回 ‘Hello World!’ 字符串。对于 web框架的使用者来说，他们并不关心如何接收 HTTP 请求，也不关心如何将请求路由到具体方法处理并将响应结果返回给用户。Web框架的使用者在大部分情况下，只需要关心如何实现业务的逻辑即可。

WSGI层

WSGI 不是服务器，也不是用于与程序交互的API，更不是真实的代码，**WSGI 只是一种接口,它只适用于 Python 语言，其全称为 Web Server Gateway Interface**，定义了 web 服务器和 web应用之间的接口规范。也就是说，只要 web服务器和 web应用都遵守WSGI 协议，那么 web服务器和 web应用就可以随意的组合。
下面的代码展示了 web服务器是如何与 web应用组合在一起的。

```
def application(env, start_response):
    start_response('200 OK', [('Content-Type', 'text/html')])
    return [b"Hello World"]
```

方法 application由 web服务器调用，参数env，start_response 由 web服务器实现并传入。其中，env是一个字典，包含了类似 HTTP_HOST，HOST_USER_AGENT，SERVER_PROTOCO 等环境变量。start_response则是一个方法，该方法接受两个参数，分别是status，response_headers。application方法的主要作用是，设置 http 响应的状态码和 Content-Type 等头部信息，并返回响应的具体结果。

上述代码就是一个完整的 WSGI 应用，当一个支持 WSGI 的 web服务器接收到客户端的请求后，便会调用这个 application 方法。WSGI 层并不需要关心env，start_response 这两个变量是如何实现的，就像在 application 里面所做的，直接使用这两个变量即可。

值得指出的是，WSGI 是一种协议，需要区分几个相近的名词：
uwsgi
同 wsgi 一样也是一种协议，uWSGI服务器正是使用了 uwsgi 协议
uWSGI
实现了 uwsgi 和 WSGI 两种协议的web服务器。注意 uWSGI 本质上也是一种 web服务器，处于上面描述的三层结构中的 web服务器层。
CGI
通用网关接口，并不限于 Python 语言，定义了 web服务器是如何向客户端提供动态的内容。例如，规定了客户端如何将参数传递给 web服务器，web服务器如何将参数传递给 web应用，web应用如何将它的输出如何发送给客户端，等等。
生产环境下的 web应用都不使用 CGI 了，CGI进程（类似 Python 解释器）针对每个请求创建，用完就抛弃，效率低下。WSGI 正是为了替代 CGI 而出现的。

说到这，我们基本理清了 WSGI 在 web服务器与 web框架之间作用：WSGI 就像一条纽带，将 web服务器与 web框架连接起来。回到本文的题目，Nginx 属于一种 web服务器，

Flask属于一种 web框架，因此，WSGI 与 Nginx、Flask 的作用就不明而喻了。

最后以 Nginx，WSGI，Flask 之间的对话结束本文。
Nginx：Hey，WSGI，我刚收到了一个请求，我需要你作些准备，然后由Flask来处理这个请求。
WSGI：OK，Nginx。我会设置好环境变量，然后将这个请求传递给Flask处理。
Flask：Thanks WSGI！给我一些时间，我将会把请求的响应返回给你。
WSGI：Alright，那我等你。
Flask：Okay，我完成了，这里是请求的响应结果，请求把结果传递给Nginx。
WSGI：Good job！Nginx，这里是响应结果，已经按照要求给你传递回来了。
Nginx：Cool，我收到了，我把响应结果返回给客户端。大家合作愉快
