一个web应用的实质就是：

- 浏览器发送一个http请求；
- 服务器收到请求生成一个HTML文档；
- 服务器把html文档作为http响应的body发送给浏览器
- 浏览器收到http响应，从http body取出HTML文档并显示。

所以，最简单的web应用就是把HTML用文件保存好，用一个现成的HTTP服务器软件，接受用户请求，从文件中读取HTML，返回。Apache、Nginx、Lighttpd等这些常见的静态服务器就是干这件事情的。

如果要动态生成HTML，就需要把上述步骤自己来实现。不过，接受HTTP请求、解析HTTP请求、发送HTTP响应都是苦力活，如果我们自己来写这些底层代码，还没开始写HTML呢，就得花个把月去读HTTP规范。

正确的做法是底层代码由专门的服务器软件实现，我们用Python专注于生成HTML文档。因为我们不希望接触到TCP连接、HTTP原始请求和响应格式，所以，需要找一个统一的接口，让我们专心用Python编写web业务。

这个接口就是WSGI：web server gateway interface。

WSGI接口定义非常简单，它要求web开发者实现一个函数，就可以响应HTTP请求。

```
def applicaton(environ, start_response):
    start_response('200 0K', [('Content-Type', 'text/html')])
    return [b'<h1>Hello,web</h1>']
```

上面applicaton()函数就是符合WSGI标准的一个HTTP处理函数，它接收两个参数：

- environ:一个包含所有HTTP请求信息的dict对象；
- start_response:一个发送HTTP响应的函数 在application()函数中，调用：

  ```
  start_response('200 0K', [('Content-Type', 'text/html')])
  ```

  就发送了HTTP响应的Header,注意Header只能发送一次，也就是只能调用一次start_response()函数。start_response()函数接收两个参数，一个是HTTP响应码，一个是一组list表示的HTTP Header，每个Header用一个包含str的tuple表示。

通常情况下，都应该把Content-Type头发给浏览器。其他很多常用的HTTP Headery也发送给浏览器。

有了WSGI，我们关心的就是如何从environ这个dict对象拿到HTTP请求信息，然后构造HTML，通过start_response()发送Header,最后返回Body。

真个application()函数本身没有涉及到任何解析HTTP的部分，也就是说，底层代码不需要我们编写，我们只负责更高层次上考虑如何响应请求就可以了。

不过，等等，这个application()函数怎么调用？如果我们自己调用，两个参数environ和start_response我们没法提供，返回的bytes也没法发给服务器。

所以application()函数必须由SWGI服务器来调用。很多符合WSGI规范的服务器，我们可以挑一个来用。但是现在，我们只想尽快测试一下我们编写的application()函数真的可以把HTML输出到浏览器，所以，要赶紧找一个简单的WSGI服务器，把我们的Web应用程序跑起来。

好消息是Python内置了一个WSGI服务器，这个模块叫wsgiref,它是用纯Python编写的WSGI服务器的参考实现。

运行WSGI服务

我们先编写Hello.py,实现web应用程序的WSGI处理函数：

```
# hello.py

def applicaton(environ, start_response):
    start_response('200 0K', [('Content-Type', 'text/html')])
    return [b'<h1>Hello,web</h1>']
```

然后，再编写一个server.py,负责启动WSGI服务器，加载applicaton()函数：

```
# server.py
# 从wsgiref模块导入
from wsgiref.simple_server import make_server
# 导入我们自己编写的applicaton函数
from hello import applicaton

httpd = make_server('', 8000, applicaton)

print('Serving HTTP on port 8000...')
# 开始监听HTTP请求
httpd.serve_forever()
```