From CGI to WSGI

Inspecting the invocation of Web application in SysAdmin approach

REN Xiaolei

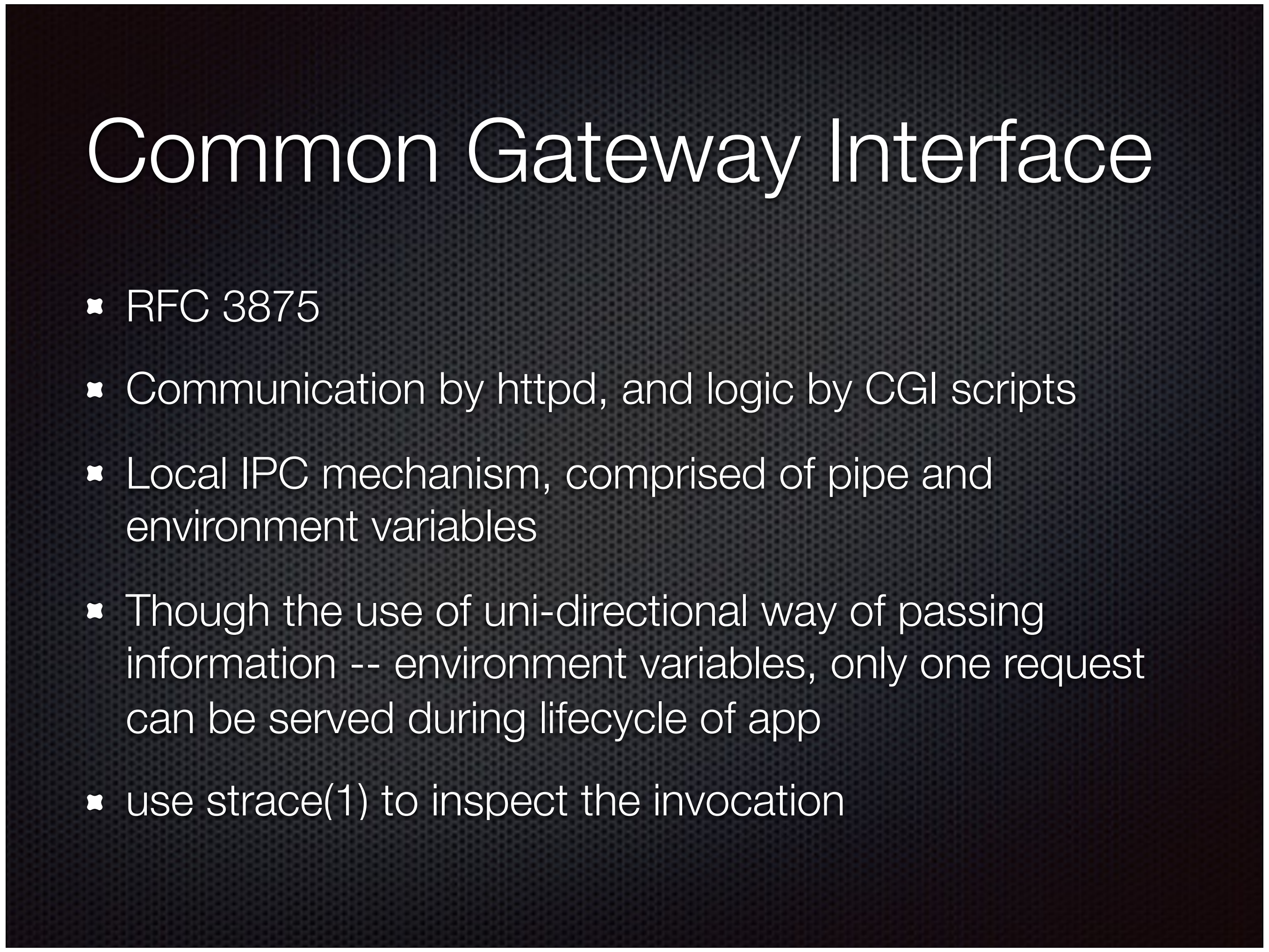
SysAdmin Team, Meituan Tech

Concepts

- Common Gateway Interface

  the historic approach
- FastCGI / SimpleCGI
- Python Web Server Gateway Interface

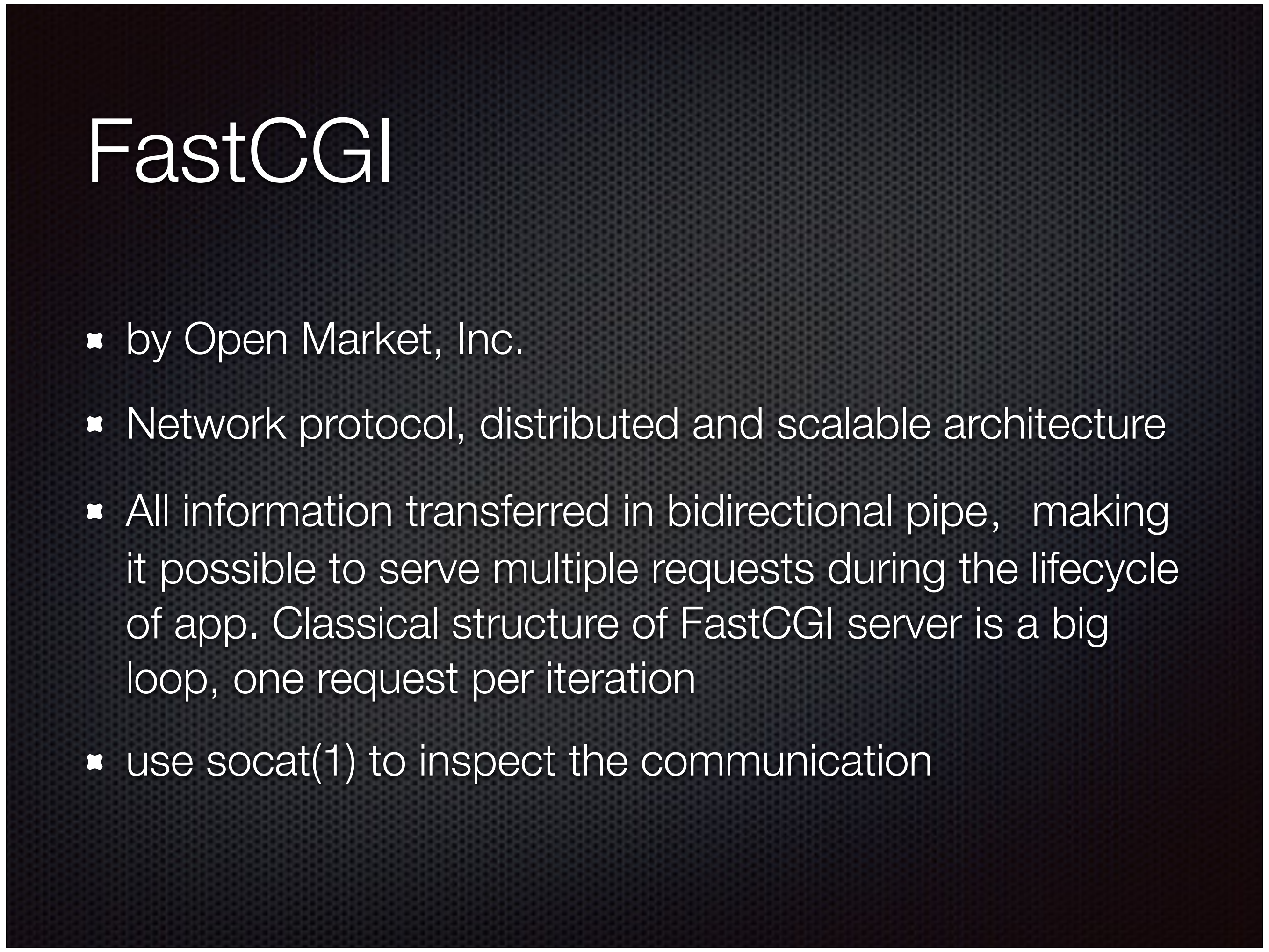
  and things like it



Common Gateway Interface

- RFC 3875
- Communication by httpd, and logic by CGI scripts
- Local IPC mechanism, comprised of pipe and environment variables
- Though the use of uni-directional way of passing information -- environment variables, only one request can be served during lifecycle of app
- use strace(1) to inspect the invocation

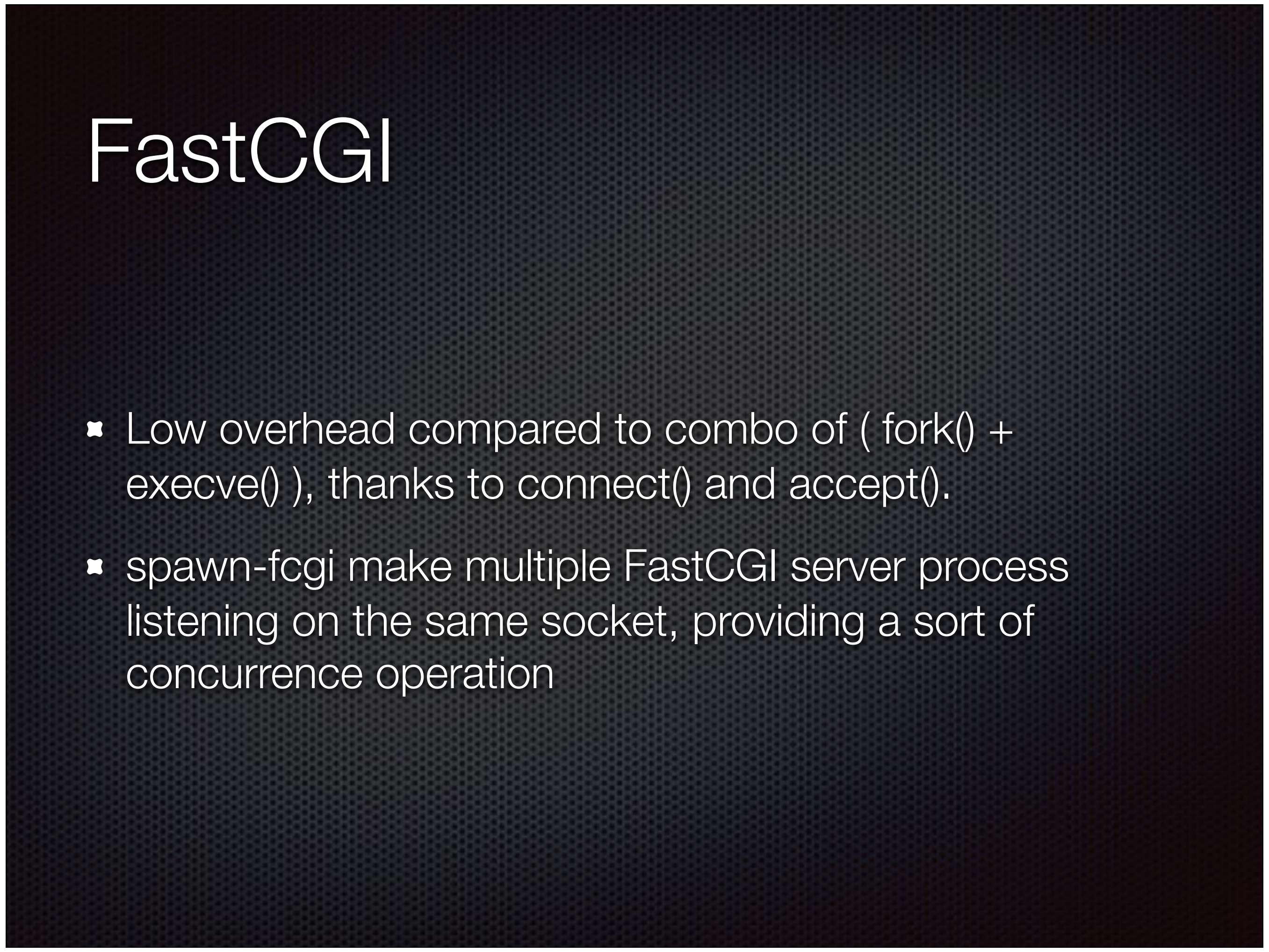
httpd fork+execve执行CGI脚本
因为环境变量只能单向、单次传递，因此下次请求无法重用上次的CGI进程。CGI脚本每次只能处理一个请求
这里插播strace检查Apache如何调用CGI程序的演示



FastCGI

- by Open Market, Inc.
- Network protocol, distributed and scalable architecture
- All information transferred in bidirectional pipe，making it possible to serve multiple requests during the lifecycle of app. Classical structure of FastCGI server is a big loop, one request per iteration
- use socat(1) to inspect the communication

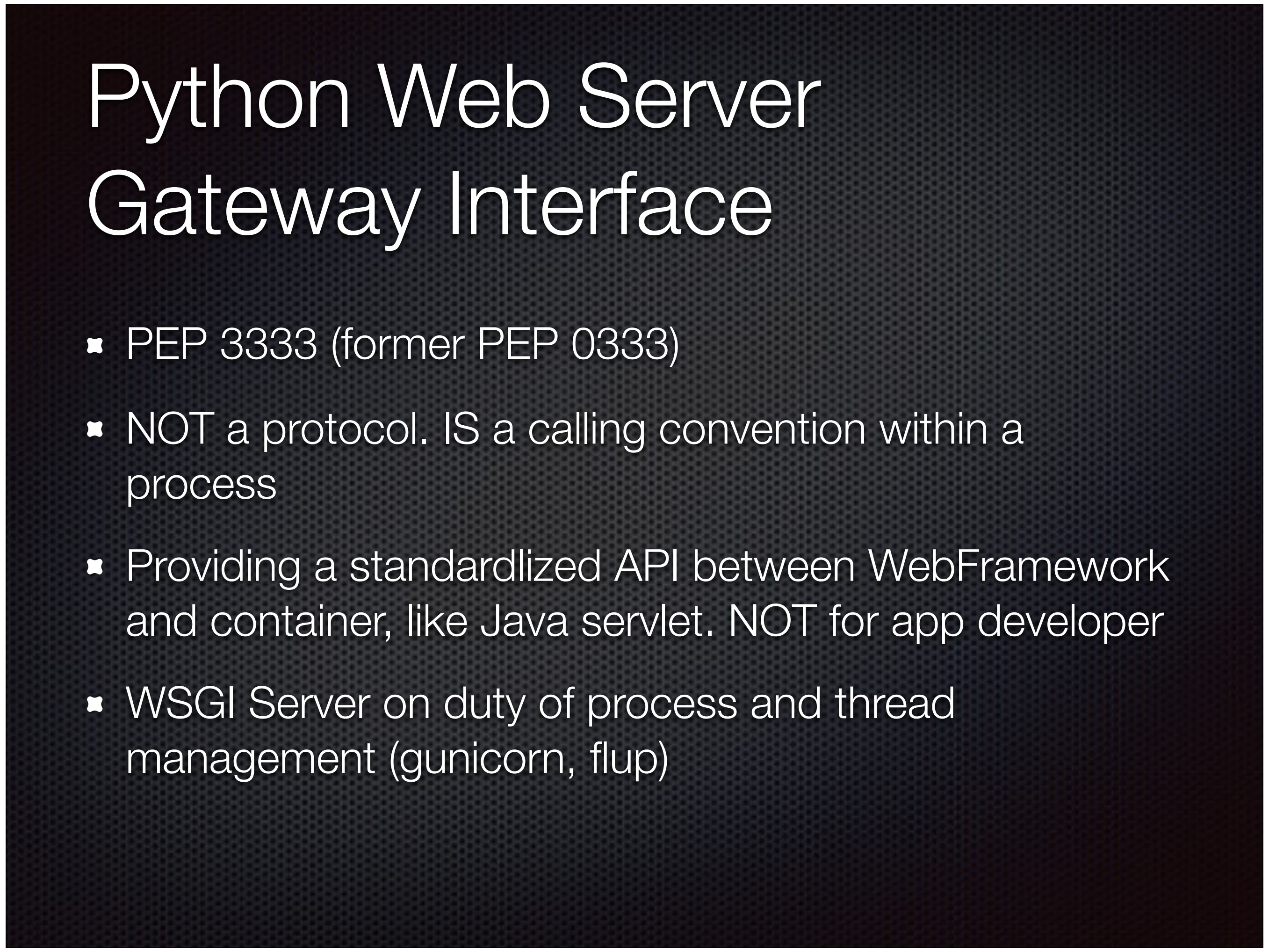
因为是网络协议，因此服务器和客户端不必在同一台机器上，方便横向扩展
所有的信息（request line、status line、headers、body）都封装成指定的格式，通过双工管道传递。没有CGI那样用环境变量传递导致的应用程序有状态的问题，因此可以在服务进程的生命周期内处理多个请求。经典的FastCGI程序架构就是一个大循环，每次循环都accept并处理一个请求
这里插播用socat检查nginx如何调用FastCGI的演示



FastCGI

- Low overhead compared to combo of (fork() + execve()), thanks to connect() and accept().
- spawn-fcgi make multiple FastCGI server process listening on the same socket, providing a sort of concurrence operation

网络通信比生成进程的开销更小
spawn-fcgi程序的详解可以看我博客文章



Python Web Server Gateway Interface

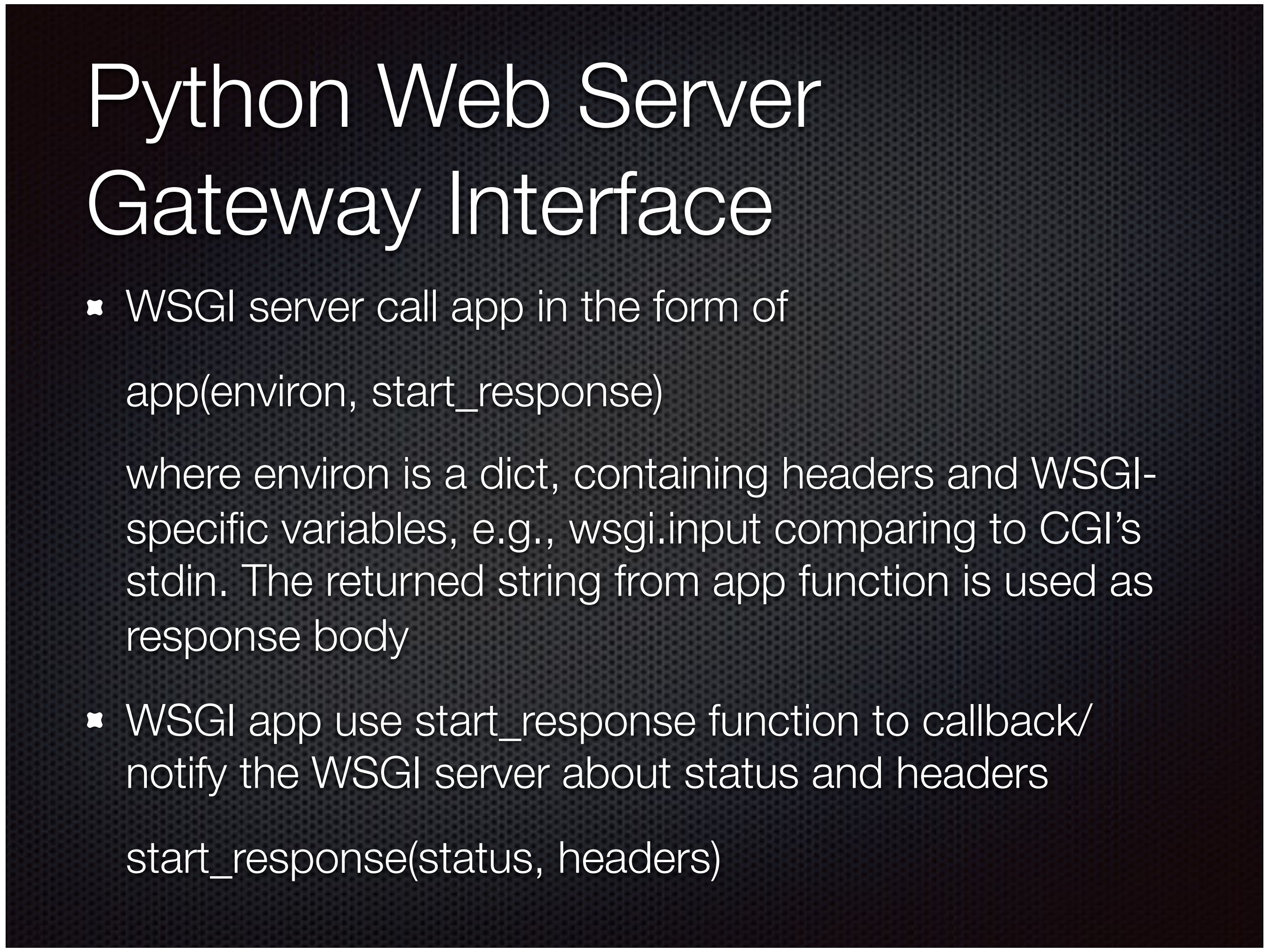
- PEP 3333 (former PEP 0333)
- NOT a protocol. IS a calling convention within a process
- Providing a standardlized API between WebFramework and container, like Java servlet. NOT for app developer
- WSGI Server on duty of process and thread management (gunicorn, flup)

注意是为框架设计的，不是为应用程序设计的。Web应用程序应该用框架开发，而不推荐直接写WSGI接口的应用程序
gunicorn支持多进程；flup支持多进程和多线程
python有select模块支持select、epoll等，因此常用的高性能WSGI server一般是用纯python写的；有些使用基于eventlet的协程机制

Python Web Server Gateway Interface

- WSGI Server on duty of communication with outside, e.g. act as a CGI script (use wsgiref.handlers.CGIHandler), act as a FastCGI server or AJP server (use flup)

  Even serve HTTP directly (wsgiref.simple_server, gunicorn, etc.)



Python Web Server Gateway Interface

- WSGI server call app in the form of

  app(environ, start_response)

  where environ is a dict, containing headers and WSGI-specific variables, e.g., wsgi.input comparing to CGI's stdin. The returned string from app function is used as response body

- WSGI app use start_response function to callback/notify the WSGI server about status and headers

  start_response(status, headers)

start_response类似于CGI中输出Headers那一段。CGI空一行之后输出的内容，在这里相当于app执行完毕后返回的字符串