2017-07-14，田宇伟

Python Web开发初识

目录

Python Web开发初识

HTTP client-server

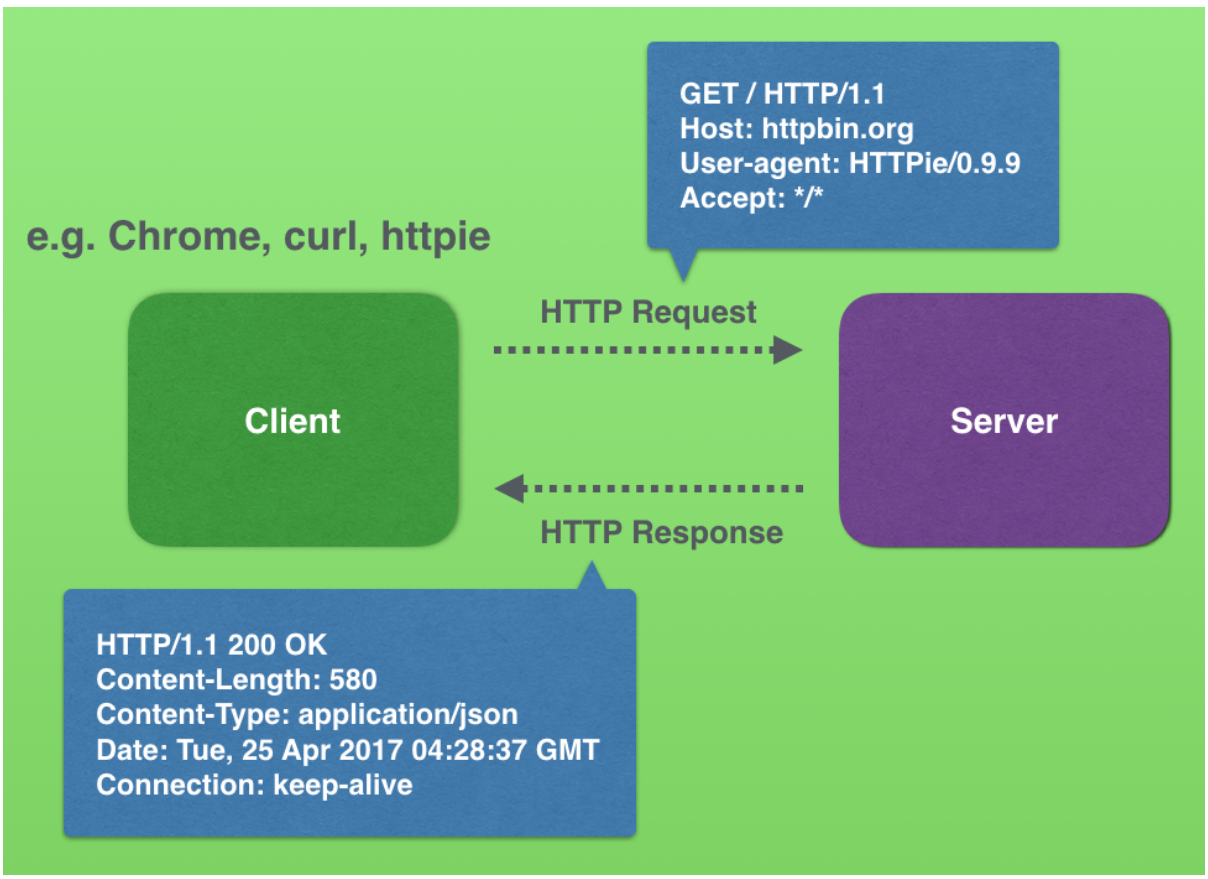


图1 HTTP Client-Server

- 通过请求和响应的交换达成通信
- 不保存通信状态（stateless）
- 使用**URI**定位互联网上的资源
- 请求资源时使用方法下达命令（GET、POST、HEAD等）
- 通过持久连接节省通信量
- 使用cookie来进行状态管理

HTTP请求

```
GET / HTTP/1.1
Connection: close
Host: httpbin.org
User-agent: HTTPie/0.9.9
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en
Accept-Charset: *, utf-8

Optional data
...
```

- 第一行定义**请求类型**、**文档（选择符）**和**协议版本**
- 接着是报头行，包括各种有关客户端的信息
- 报头行后面是一个空白行，表示报头行结束
- 之后是发送表单的信息或者上传文件的事件中可能出现的数据
- 报头的每一行都应该使用回车符或者换行符（'\r\n'）终止

下表是常见HTTP请求方法：

表1 HTTP常见请求方法

方法	描述
GET	获取文档
POST	将数据发布到表单
HEAD	仅返回报头信息
PUT	将数据上传到服务器
...	...

HTTP响应

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 580
Content-Type: application/json
Date: Tue, 25 Apr 2017 04:28:37 GMT
Server: gunicorn/19.7.1
...
Header: data

Data
...
```

- 第一行表示**HTTP协议版本**、**成功代码**和**返回消息**
- 响应行之后是一系列报头字段，包含返回文档的类型、文档大小、Web服务器软件、cookie等方面的信息
- 通过空白行结束报头
- 之后是所请求文档的原始数据

下表是HTTP常见状态码：

表2 HTTP常见状态码

代码	描述	符号常量
成功代码（2xx）		
200	成功	OK
201	创建	CREATED
202	接受	ACCEPTED
204	无内容	NO_CONTENT
重定向（3xx）		
300	多种选择	MULTIPLE_CHOICES
301	永久移动	MOVED_PERMANENTLY
302	可被303替代	FOUND
303	临时移动	SEE_OTHER
304	不修改	NOT_MODIFIED
客户端错误（4xx）		
400	请求错误	BAD_REQUEST
401	未授权	UNAUTHORIZED
403	禁止访问	FORBIDDEN
404	未找到	NOT_FOUND
405	方法不允许	METHOD_NOT_ALLOWED
服务器错误（5xx）		
500	内部服务器错误	INTERNAL_SERVER_ERROR
501	未实现	NOT_IMPLEMENTED
502	网关错误	BAD_GATEWAY
503	服务不可用	SERVICE_UNAVAILABLE

Python3 标准Web库

- `http` 处理所有客户端—服务器HTTP请求的具体细节
  - `client` 处理客户端部分
  - `server` 提供了实现HTTP服务器的各种类
  - `cookies` 支持在服务器端处理HTTP cookie
  - `cookiejar` 支持在客户端存储和管理HTTP cookie
- `urllib` 基于 `http` 的高层库，用于编写与HTTP服务器等交互的客户端
  - `request` 处理客户端请求
  - `response` 处理服务器端响应
  - `parse` 用于操作URL字符串

```
import urllib.request as ur

url = 'http://httpbin.org/'
conn = ur.urlopen(url)
print(conn)
print('=' * 50)
data = conn.read()
print(data[:16])
print(conn.status)
```

```
print(conn.getheader('Content-Type'))
for key, value in conn.getheaders():
    print(key, value, sep=': ')
```

最简单的Python Web服务器

```
python -m http.server
```

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [26/Apr/2017 09:50:56] "GET / HTTP/1.1" 200 -
...
```

第三方工具和库

HTTPie

https://httpie.org

HTTPie（读aych-tee-tee-pie）是一个 HTTP 的命令行客户端。其目标是让 CLI 和 web 服务之间的交互尽可能的人性化。

这个工具提供了简洁的 `http` 命令，允许通过自然的语法发送任意 HTTP 请求数据，展示色彩化的输出。

HTTPie 可用于与 HTTP 服务器做测试、调试和常规交互。

HTTPie 用 Python 编写，用到了 Requests 和 Pygments 这些出色的库。

httpbin

http://httpbin.org

使用 Python + Flask 编写的 HTTP 请求和响应服务。

Installation

Run it as a WSGI app using Gunicorn:

```
$ pip install httpbin
$ gunicorn httpbin:app
```

```
$ http http://httpbin.org/user-agent
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 35
Content-Type: application/json
Date: Wed, 26 Apr 2017 04:32:28 GMT
Server: gunicorn/19.7.1
Via: 1.1 vegur

{
    "user-agent": "HTTPie/0.9.9"
}
```

Requests库

> **HTTP for Humans.**

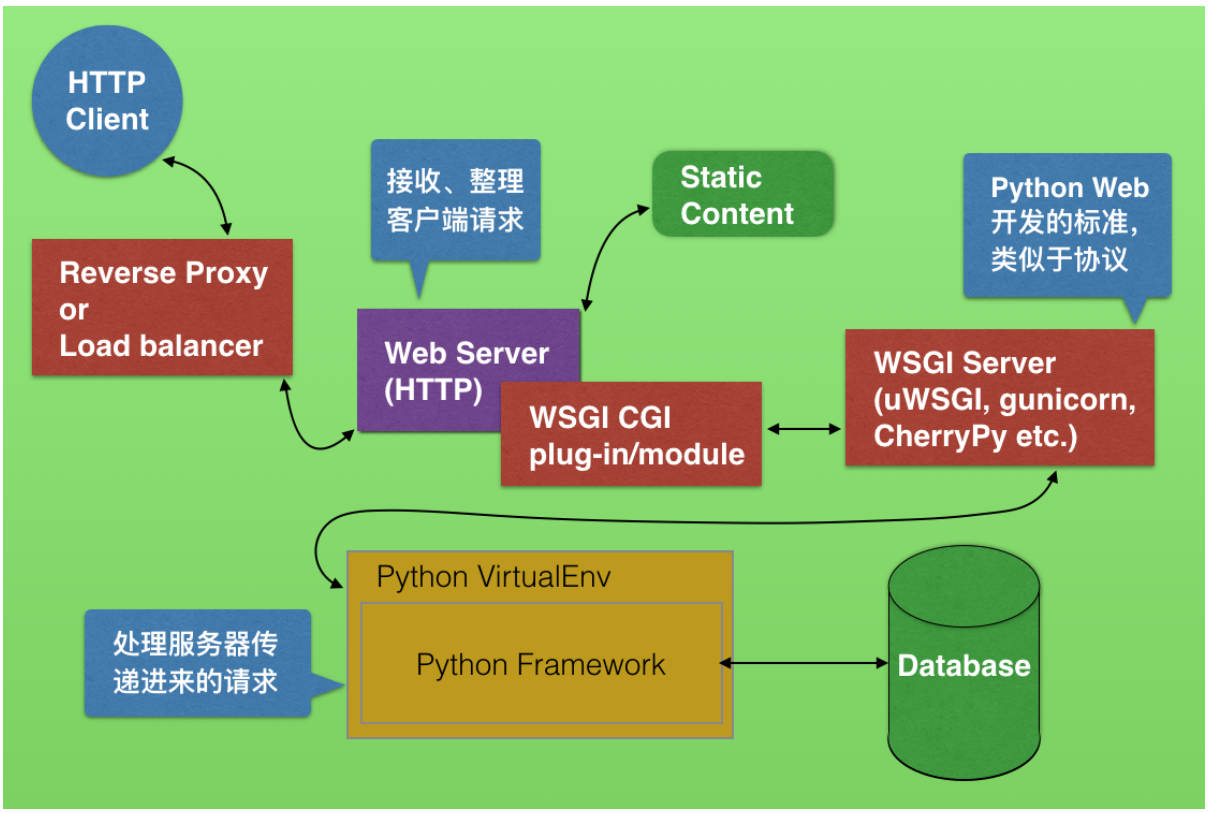
Docs: http://docs.python-requests.org/en/master/

Repo: https://github.com/kennethreitz/requests

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...'
>>> r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

```
import requests
s = requests.Session()

print(s.get('http://httpbin.org/ip').text)

print(s.get('http://httpbin.org/get').json())

print(s.post('http://httpbin.org/post',
             {'key':'value'},
             headers={'user-agent':'httpie'}).text)

print(s.get('http://httpbin.org/status/404').status_code)

print(s.get('http://httpbin.org/html').text)

print(s.get('http://httpbin.org/deny').text)
```

简单Python Web架构



实际生产中，Python程序是放在服务器的 HTTP Server（比如 Apache，Nginx 等）上的。

服务器程序怎么把接受到的请求传递给Python?

怎么在网络的数据流和Python的结构体之间转换?

处理上面两项工作就是图中 WSGI Server 做的事情。

Python WSGI

WSGI（Web Server Gateway Interface）是一套关于程序端和服务器端的规范，或者说统一的接口。

先看一下面向 HTTP 的 Python Web程序需要关心的内容：

- 请求
  - **请求的方法 method**
  - **请求的地址 url**
  - **请求的内容**
  - 请求的头部 header
  - 请求的环境信息
- 响应
  - **状态码 status_code**
  - **响应的数据**
  - 响应的头部

WSGI的任务就是**把上面的数据在 HTTP Server 和 Python 程序之间简单友好地传递。**它是一个标准，被定义在PEP 333。需要 HTTP Server 和 Python 程序都要遵守一定的规范，实现这个标准的约定内容，才能正常工作。

Web框架有什么

一个Web框架，至少要具备处理客户端请求和服务端响应的能力。

路由
: 解析URL并找到对应的服务端文件或者Python服务器代码。

模板
: 把服务端数据合并成HTML页面。

认证和授权
: 处理用户名、密码和权限。

Seesion
: 处理用户在多次请求之间需要存储的数据。

开发环境及工具

- **开发环境**

  Ubuntu 16.04 LTS

- **Visual Studio Code**
  - 拓展：Python, MagicPython
  - 用户配置文件

日期：2017-07-14 作者：田宇伟