Python Web开发初识

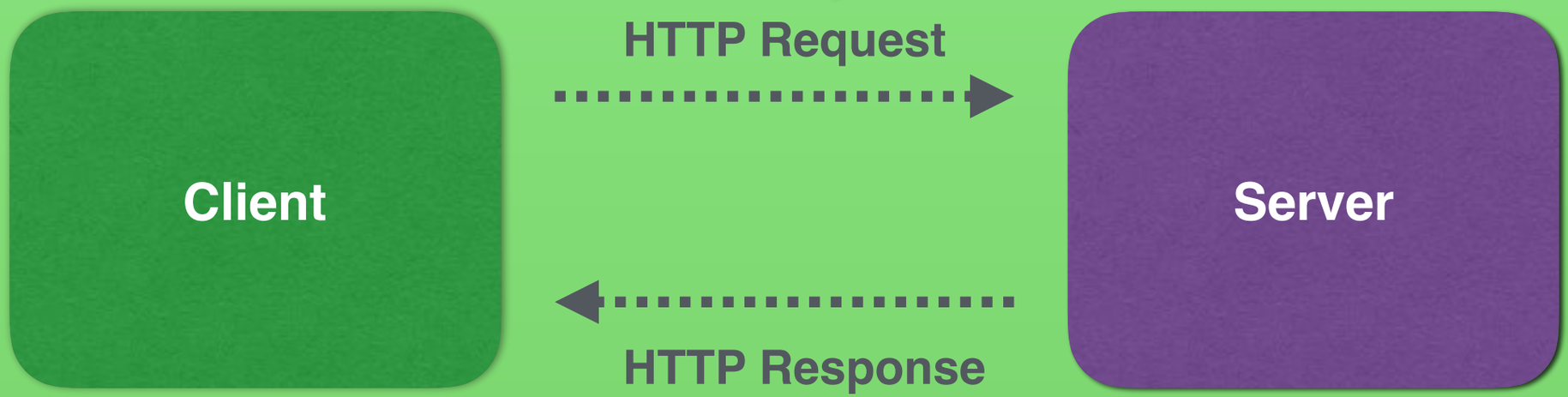
田宇伟

Apr 26, 2017

HTTP client-server

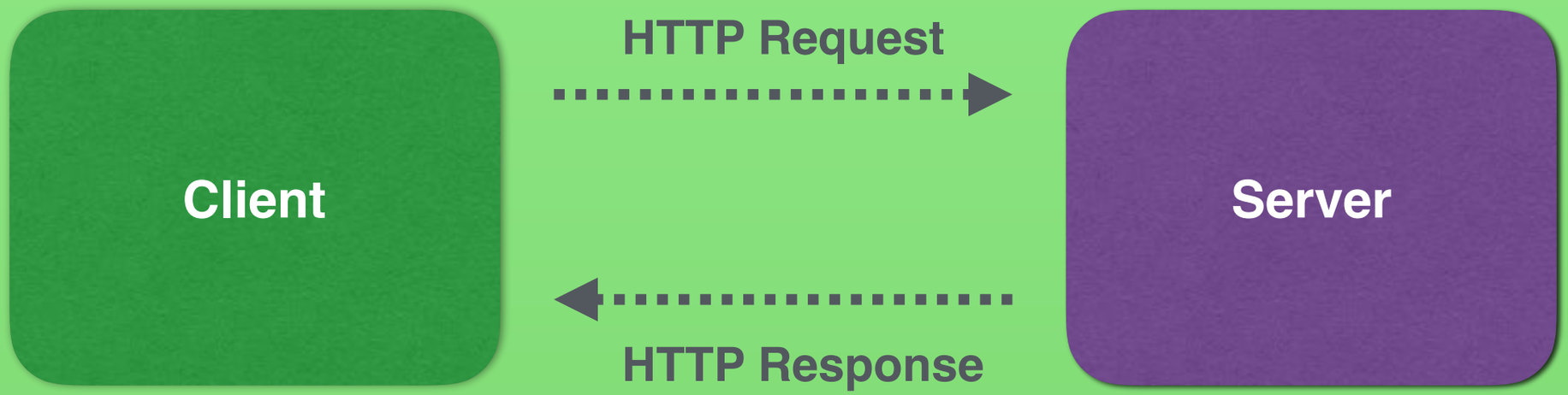
e.g. Chrome, curl, httpie

GET / HTTP/1.1
Host: httpbin.org
User-agent: HTTPie/0.9.9
Accept: */*



HTTP/1.1 200 OK
Content-Length: 580
Content-Type: application/json
Date: Tue, 25 Apr 2017 04:28:37 GMT
Connection: keep-alive

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

HTTPie

https://github.com/jakubroztocil/httpie

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

HTTPie

- 直观的语法
- 格式化和色彩化的终端输出
- 内置 JSON 支持
- 支持上传表单和文件
- HTTPS、代理和认证
- 任意请求数据
- 自定义头部
- 持久性会话
- 类 Wget 下载
- 支持 Python 2.6, 2.7 和 3.x
- 支持 Linux, Mac OS X 和 Windows
- 插件
- 文档
- 测试覆盖率

http请求报头

```
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

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

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
…	…

常见HTTP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

httpbin

使用 Python + Flask 编写的 HTTP 请求和响应服务

http://httpbin.org

https://github.com/kennethreitz/httpbin

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"
}
```

httpbin

httpbin 示例

```
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)
```

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

Python3 标准Web库

使用 urllib 获取网页内容

```
import urllib.request as ur

url = 'http://httpbin.org/'
conn = ur.urlopen(url)
print(conn)
# <http.client.HTTPResponse object at 0x1089659b0>
```

Python3 标准Web库

使用 urllib 获取网页内容

```
data = conn.read()
print(data[:16])
print(conn.status)

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

```
b'<!DOCTYPE html>\n'
200
text/html; charset=utf-8
Connection: close
Server: gunicorn/19.7.1
Date: Wed, 26 Apr 2017 00:40:50 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 12373
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Via: 1.1 vegur
```

Requests 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, ...}
```

Requests HTTP for Humans

- 国际化域名和 URL
- Keep-Alive & 连接池
- 带持久 Cookie 的会话
- 浏览器式的 SSL 认证
- 基本/摘要式的身份认证
- 优雅的 key/value Cookie
- 自动解压
- 自动内容解码
- Unicode 响应体
- 文件分块上传
- 连接超时
- 流下载
- 分块请求
- 线程安全
- 支持 Python 2.6—3.5，而且能在PyPy下完美运行

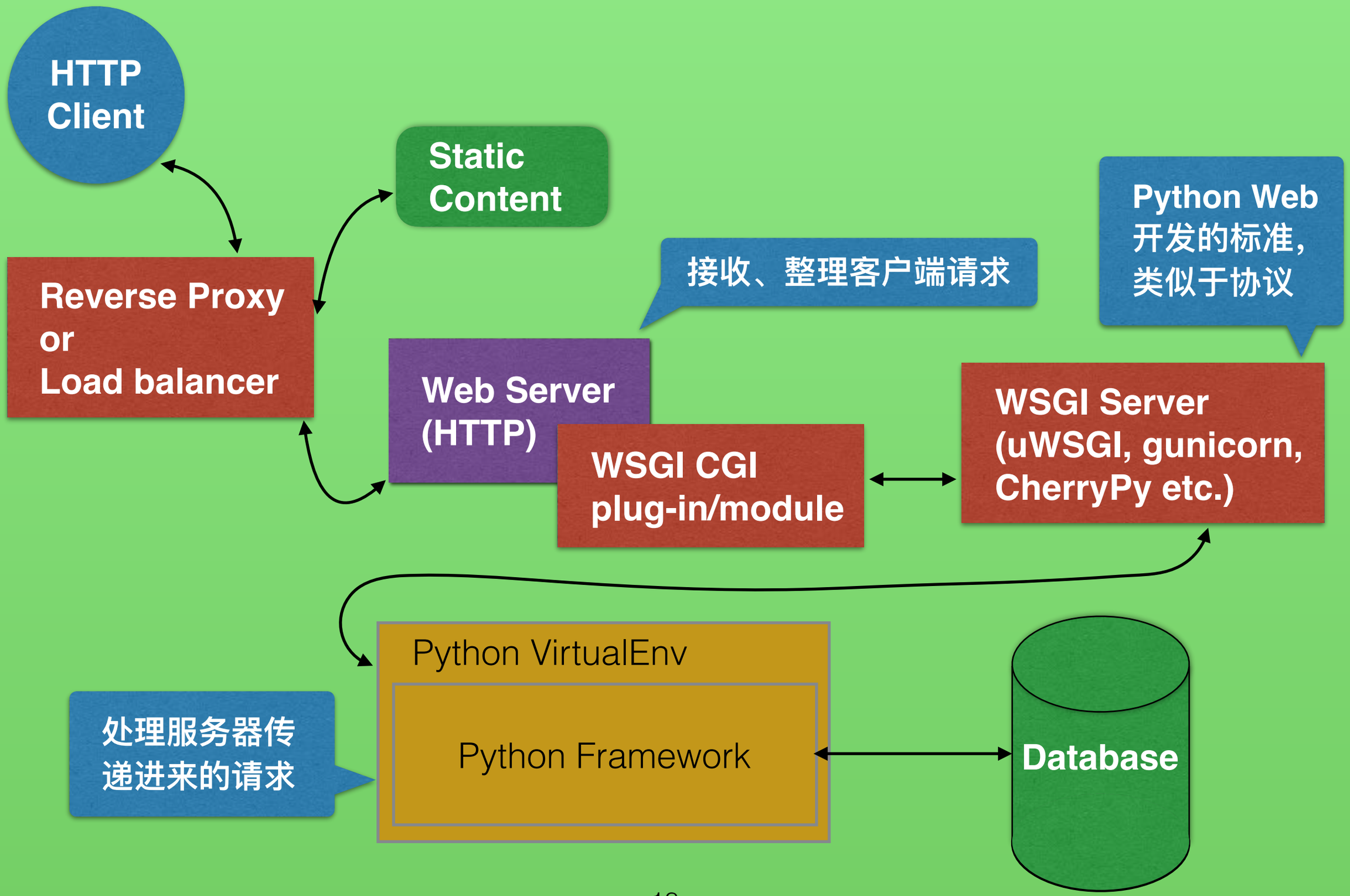
Python Web Server

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 -
...
```

WSGI Web Server Gateway Interface



Python Web框架

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。

Python Web框架

http://bottlepy.org

http://webpy.org

http://flask.pocoo.org

https://trypyramid.com

http://www.tornadoweb.org

https://www.djangoproject.com

http://web2py.com

And more ...

Web API 和 REST

Representational State Transfer

符合REST架构定义的特征

- 客户端-服务器

  客户端和服务器之间必须有明确的界线。

- 无状态

  客户端发出的请求中必须包含所有必要的信息。服务器不能在两次请求之间保存客户端的任何状态。

- 缓存

  服务器发出的响应可以标记为可缓存或不可缓存，这样出于优化目的，客户端(或客户端和服务器之间的中间服务)可以使用缓存。

Web API 和 REST

符合REST架构定义的特征

- 接口统一

  客户端访问服务器资源时使用的协议必须一致，定义良好，且已经标准化。REST Web 服务最常使用的统一接口是 HTTP 协议。

- 系统分层

  在客户端和服务器之间可以按需插入代理服务器、缓存或网关，以提高性能、稳定性和伸缩性。

- 按需代码

  客户端可以选择从服务器上下载代码，在客户端的环境中执行。

Web API 和 REST

REST架构API中使用的HTTP请求方法

请求方法	目 标	说 明	HTTP状态码
GET	单个资源的URL	获取目标资源	200
GET	资源集合的URL	获取资源的集合(如果服务器实现了分页，就是一页中的资源)	200
POST	资源集合的URL	创建新资源，并将其加入目标集合。服务器为新资源指派URL，并在响应的Location首部中返回	201
PUT	单个资源的URL	修改一个现有资源。如果客户端能为资源指派URL，还可用来创建新资源	200
DELETE	单个资源的URL	删除一个资源	200
DELETE	资源集合的URL	删除目标集合中的所有资源	200

Web API 和 REST

REST Web 服务常用编码方式

- JSON（JavaScript Object Notation, JavaScript对象表示法）
- XML（Extensible Markup Language, 可扩展标记语言）

示例：一篇博客文章对应的资源

```
{
    "url": "http://www.example.com/api/posts/12345",
    "title": "Writing RESTful APIs in Python",
    "author": "http://www.example.com/api/users/2",
    "body": "... text of the article here ...",
    "comments": "http://www.example.com/api/posts/12345/comments"
}
```

Thanks!