

[illegible]

刘煜

## 西安中心

# 目录

HTTP协议概述

HTTP消息结构

HTTP请求方法

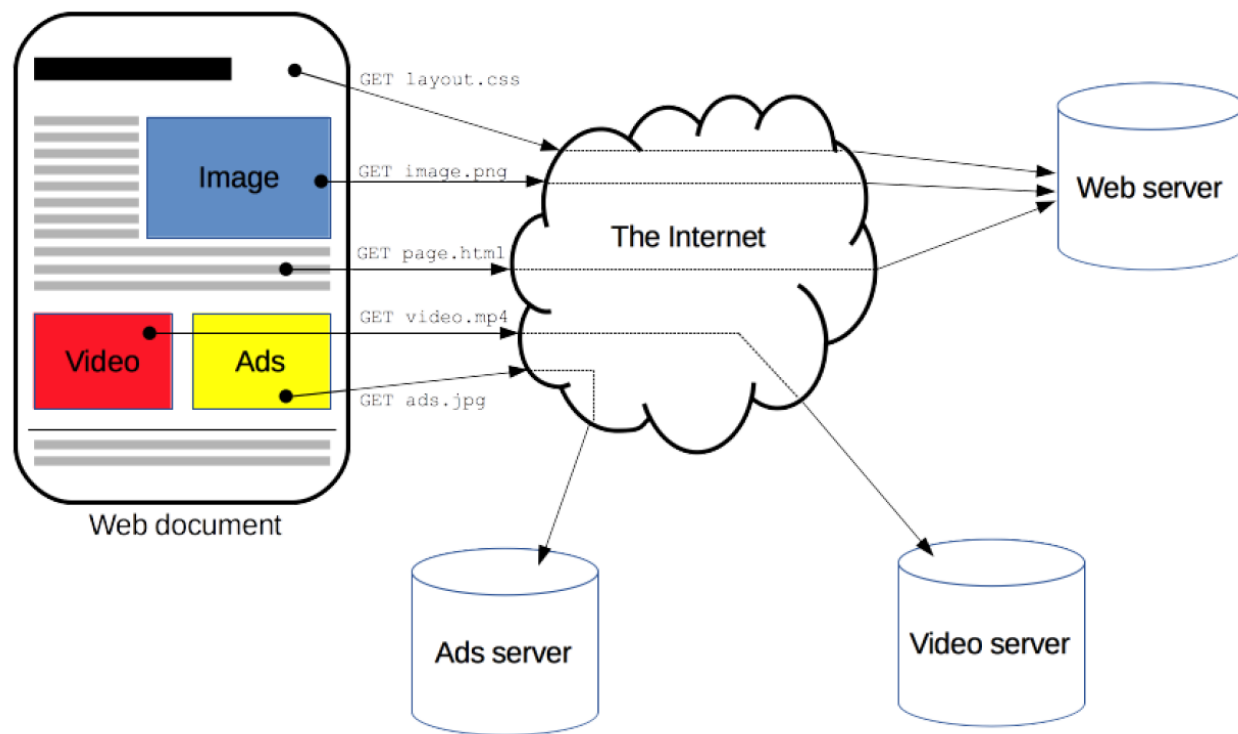
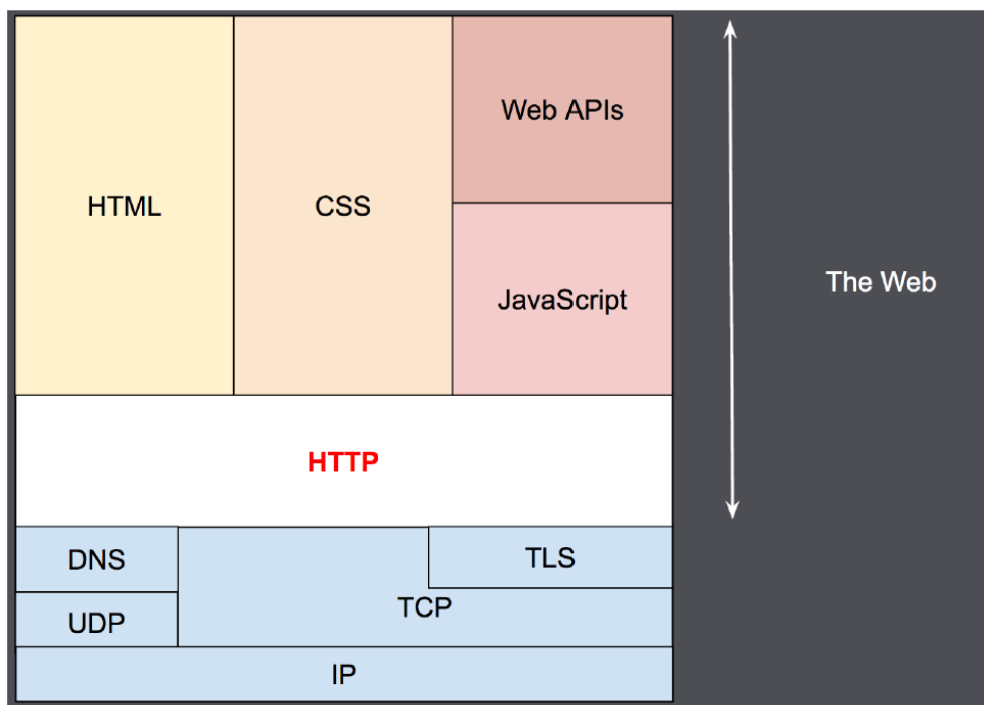
HTTP响应头信息

HTTP状态码

HTTP内容类型

# | 概述

HTTP是一种能够获取如 HTML 这样的网络资源的通讯协议。它是在 Web 上进行数据交换的基础，是一种 client-server 协议，由像浏览器这样的客户端发出的消息叫做 requests，被服务端响应的消息叫做 responses。一个完整的 Web 文档通常是由不同的子文档拼接而成的，像是文本、布局描述、图片、视频、脚本等等。

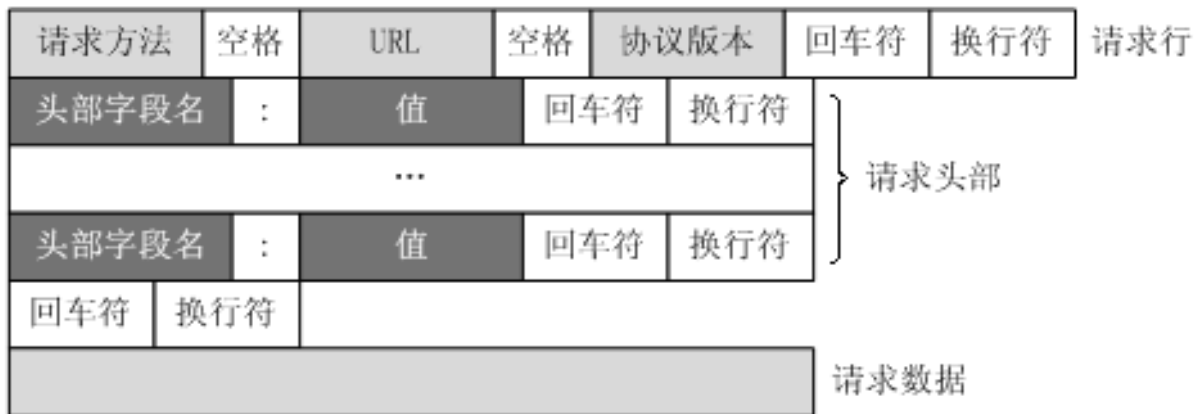
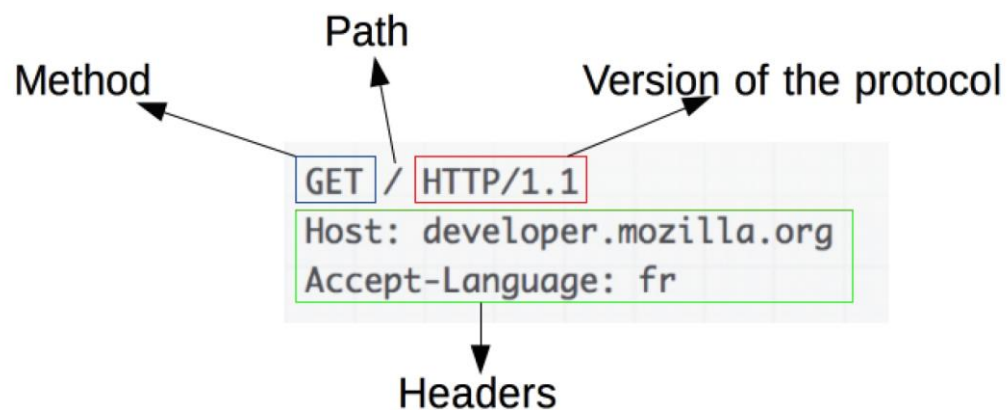


## 协议基础

- HTTP是基于客户端/服务端（C/S）的架构模型，通过一个可靠的链接来交换信息，是一个无状态请求/响应协议。
- 一个HTTP"客户端"是一个应用程序（Web浏览器或其他任何客户端），通过连接到服务器达到向服务器发送一个或多个HTTP的请求的目的。
- 一个HTTP"服务器"同样也是一个应用程序（通常是一个Web服务，如Apache Web服务器或IIS服务器等），通过接收客户端的请求并向客户端发送HTTP响应数据。
- HTTP使用统一资源标识符（Uniform Resource Identifiers, URI）来传输数据和建立连接。
- 一旦建立连接后，数据消息就通过类似Internet邮件所使用的格式[RFC5322]和多用途Internet邮件扩展（MIME）[RFC2045]来传送。

# 客户端请求消息

客户端发送一个HTTP请求到服务器的请求消息包括以下格式：请求行（request line）、请求头部（header）、空行和请求数据四个部分组成，下图给出了请求报文的一般格式。



## 服务器响应消息

HTTP响应也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。

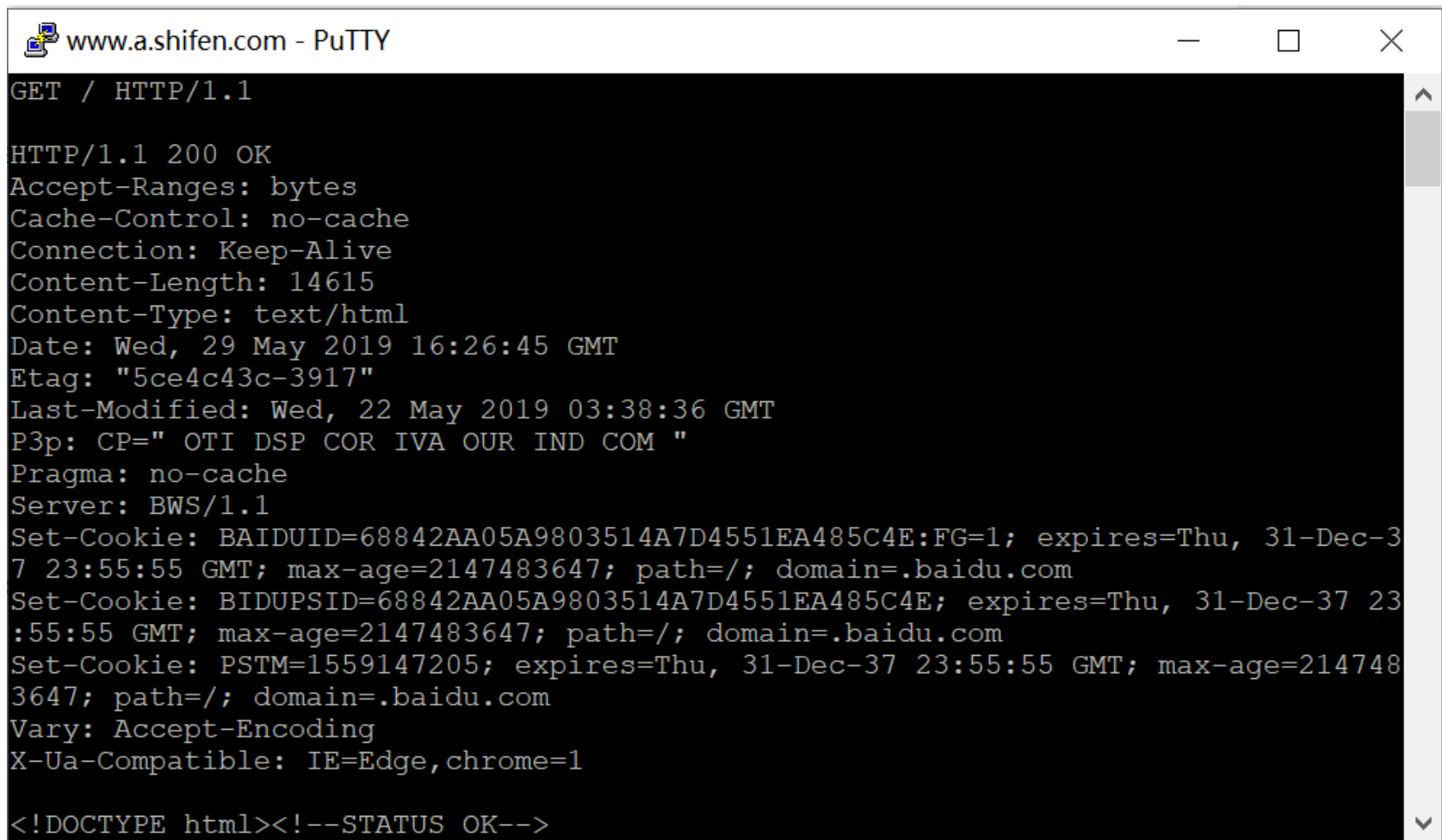
The diagram illustrates the structure of an HTTP response. It shows a sequence of lines representing the response, with labels on the right side connected by lines to the corresponding parts of the response:

- 状态行** (Status Line): Points to the first line, `HTTP/1.1 200 OK`.
- 消息报头** (Message Header): Points to the next three lines: `Date: Sat, 31 Dec 2005 23:59:59 GMT`, `Content-Type: text/html; charset=ISO-8859-1`, and `Content-Length: 122`.
- 空行** (Blank Line): Points to the empty line between the headers and the body.
- 下面的就是响应正文了** (The following is the response body): Points to the HTML content starting from `<html>` and ending at `</html>`.

```
HTTP/1.1 200 OK
Date: Sat, 31 Dec 2005 23:59:59 GMT
Content-Type: text/html; charset=ISO-8859-1
Content-Length: 122

<html>
<head>
<title>Wrox Homepage</title>
</head>
<body>
<!-- body goes here -->
</body>
</html>
```

## 实例



```
www.a.shifen.com - PuTTY
GET / HTTP/1.1

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: no-cache
Connection: Keep-Alive
Content-Length: 14615
Content-Type: text/html
Date: Wed, 29 May 2019 16:26:45 GMT
Etag: "5ce4c43c-3917"
Last-Modified: Wed, 22 May 2019 03:38:36 GMT
P3p: CP=" OTI DSP COR IVA OUR IND COM "
Pragma: no-cache
Server: BWS/1.1
Set-Cookie: BAIDUID=68842AA05A9803514A7D4551EA485C4E:FG=1; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: BIDUPSID=68842AA05A9803514A7D4551EA485C4E; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Set-Cookie: PSTM=1559147205; expires=Thu, 31-Dec-37 23:55:55 GMT; max-age=2147483647; path=/; domain=.baidu.com
Vary: Accept-Encoding
X-Ua-Compatible: IE=Edge,chrome=1

<!DOCTYPE html><!--STATUS OK-->
```

# | 请求方法

根据HTTP标准，HTTP请求可以使用多种请求方法。

HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。

HTTP1.1新增了五种请求方法： OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。



## 响应头信息

HTTP请求头提供了关于请求，响应或者其他的发送实体的信息。

应答头	说明
Allow	服务器支持哪些请求方法（如GET、POST等）。
Content-Encoding	文档的编码（Encode）方法。只有在解码之后才可以得到Content-Type头指定的内容类型。
Content-Length	表示内容长度。
Content-Type	表示后面的文档属于什么MIME类型。
Date	当前的GMT时间。

# | 状态码

当浏览者访问一个网页时，浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头（server header）用以响应浏览器的请求。

常见的HTTP状态码：

200 - 请求成功

301 - 资源（网页等）被永久转移到其它URL

404 - 请求的资源（网页等）不存在

500 - 内部服务器错误

分类	分类描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

# | 内容类型

Content-Type，内容类型，一般是指网页中存在的Content-Type，用于定义网络文件的类型和网页的编码，决定浏览器将以什么形式、什么编码读取这个文件。**MIME**(Multipurpose Internet Mail Extensions)多用途互联网邮件扩展类型。

文件扩展名	Content-Type(Mime-Type)	文件扩展名	Content-Type(Mime-Type)
*（二进制流，不知道下载文件类型）	application/octet-stream	.tif	image/tiff
.001	application/x-001	.301	application/x-301
.323	text/h323	.906	application/x-906
.907	drawing/907	.a11	application/x-a11
.acp	audio/x-mei-aac	.ai	application/postscript
.aif	audio/aiff	.aifc	audio/aiff
.aiff	audio/aiff	.anv	application/x-anv

## RESTful API

表现层状态转换（Representational State Transfer，缩写：REST）是Roy Thomas Fielding博士于2000年在他的博士论文中提出来的一种互联网软件架构风格。用开发网站的方式实现远程接口调用，网站即软件。REST通常基于使用HTTP，URI，和XML以及HTML这些现有的广泛流行的协议和标准。

特点：

- 资源是由URI来指定。
- 对资源的操作包括获取、创建、修改和删除资源，这些操作正好对应HTTP协议提供的GET、POST、PUT和DELETE方法。
- 通过操作资源的表现形式来操作资源。
- 资源的表现形式则是XML或者JSON，客户端软件可以是浏览器或者手机APP，甚至是其他服务。

## REST API例子

一个简单的网络商店应用，列举所有商品：

GET <http://www.store.com/products>

呈现某一件商品：

GET <http://www.store.com/products/12345>

下单购买：

POST <http://www.store.com/orders>

<purchase-order>

<item> ... </item>

</purchase-order>

## REST API的优点

- 可更高效利用缓存来提高响应速度
- 通讯本身的无状态性可以让不同的服务器的处理一系列请求中的不同请求，提高服务器的扩展性
- 浏览器即可作为客户端，简化软件需求
- 相对于其他叠加在HTTP协议之上的机制，REST的软件依赖性更小
- 不需要额外的资源发现机制
- 在软件技术演进中的长期的兼容性更好



海量视频 贴身学习



超多干货 实时更新

# THANKS

— 谢谢 —