

HTTP报文前言

- HTTP基于TCP/IP通信协议来传递数据
 - HTTP基于客户端/服务端（C/S）架构模型
 - 通过一个可靠的链接来交换信息，是一个无状态的请求/响应协议
 - 限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间
 - 只要客户端和服务端知道如何处理的数据内容，任何类型的数据都可以通过HTTP发送。客户端以及服务器指定使用适合的MIME-type内容类型
 - Multipurpose Internet Mail Extensions type 多用途互联网邮件扩展类型
- 报文：客户端和服务端之间的信息传递
 - 无状态的请求/响应协议：断开后都是重新连接，没有之前连接的记忆
 - 限制每次连接只处理一个请求：只是传统的TCP/IP处理方式

- 协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快
- HTTP报文定义：在客户端与服务器之间发送的数据块。这些数据块以一些文本的元信息（meta-information）开头，描述了报文的内容及含义，后面跟着可选的数据部分，这些报文在客户端、服务器和代理之间流动。所以HTTP报文的发送也叫报文流。（[前端开发之JavaScript深度指南【JS++前端】](#)展示）
- 每条HTTP报文包含一个客户端请求和服务端响应
- （请求报文Request和响应报文Response）

HTTP报文

HTTP报文

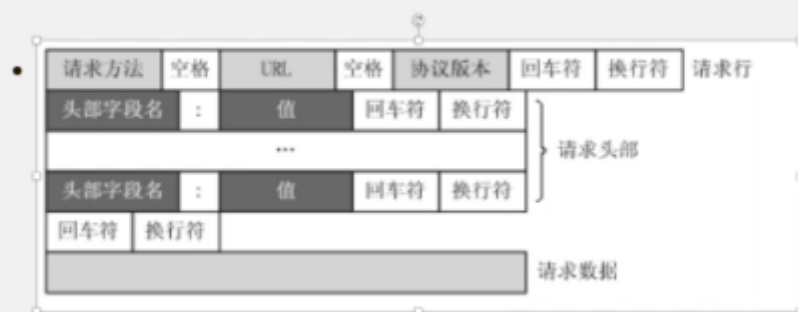


HTTP报文

- 报文的组成部分:
- 1、对报文进行描述的起始行
- 2、包含属性的首部/头部 (header)
- 3、包含数据的主体 (body) (可选项)



• HTTP报文基本格式



HTTP报文- GENERAL（通用报）

▼ General

Request URL: <https://www.baidu.com/>

Request Method: GET

Status Code: 200 OK

Remote Address: 180.97.33.108:443

Referrer Policy: unsafe-url

HTTP报文-（请求体）

▼ Form Data

[view source](#)

[view URL encoded](#)

lang: all

field: wk

kind: all

way: new

page: 1

post

▼ Query String Parameters

[view source](#)

[view URL encoded](#)

context: {"nid":"news_9015576141585109101"}

n_type: 0

p_from: 1

get

HTTP报文-请求方式

- 8种
- GET/POST
- OPTIONS: 返回服务器针对特定资源所支持的HTTP请求方法
- HEAD: 返回与GET请求相一致的响应，响应体被返回
- PUT: 向指定资源位置上传其最新内容 (form表单不支持)
- DELETE: 请求服务器删除Request-URI所标识的资源 (form表单不支持)
- TRACE: 回显服务器收到的请求，主要用于测试或诊断
- CONNECT: 连接改为管道方式的代理服务器

- put: 上传资源, form表单不支持、提交即存储的原则 (无验证机制, 安全漏洞)、需配置服务器支持put方式转发打给后端操作。
- delete: 删除资源, form表单不支持、提交即删除的原则 (无验证机制, 安全漏洞)、需配置服务器支持put方式转发打给后端操作。
- post: 修改资源
- get: 获取资源
- 对应的就是数据的增、删、改、查
- 4种不同的请求方式是为了分清楚不同请求的目的, 但是并不代表用了post就一定要修改数据, 用get就不能修改资源。

- GET/POST请求方式
- 案例展示
- form表单 GET/POST方式提交数据 (AJAX提交数据)
- 总结: GET主要用来获取数据
- GET的数据在请求体中是查询字符串参数 (Query String Parameters)
- POST主要用于传输数据到后端进行增加、删除、更新数据、提交表单
- POST的数据在请求体中是表单数据 (Form Data)
- GET/POST: view source中仍然是url参数键值对形式 a=1&b=2

- 实际在传输数据时: GET/POST 都是用 a=1&b=2 这种形式传数据的, 只是get在url中提现

- GET/POST请求方式

- 1、POST更安全

- 不会作为url的一部分、不会被缓存、保存在服务器日志和浏览器记录中

- 2、POST发送的数据量更大（GET有url长度限制）

- 长度限制：IE（2083字节） firefox(65536字符) chrome(8182字符) safari(80000字符) opera(190000字符)

- 3、POST能发送更多的数据类型（各种类型的文件）

- GET只能发送ASCII字符

- GET/POST请求方式

- 4、POST比GET速度慢

- 2、POST接收数据之前会先将请求头发送给服务器确认，然后发送数据

- POST过程：

- 1、第三次握手，浏览器确认并发送post请求头
- 2、服务器返回状态码100后，continue响应
- 3、浏览器开始发送数据
- 4、服务器返回200 Ok响应

- GET/POST请求方式

- 4、POST比GET速度慢

- 2、POST接收数据之前会先将请求头发送给服务器确认，然后发送数据

- GET过程：

- 1、第三次握手，浏览器确认并发送请求头和数据
- 2、服务器返回200 Ok响应

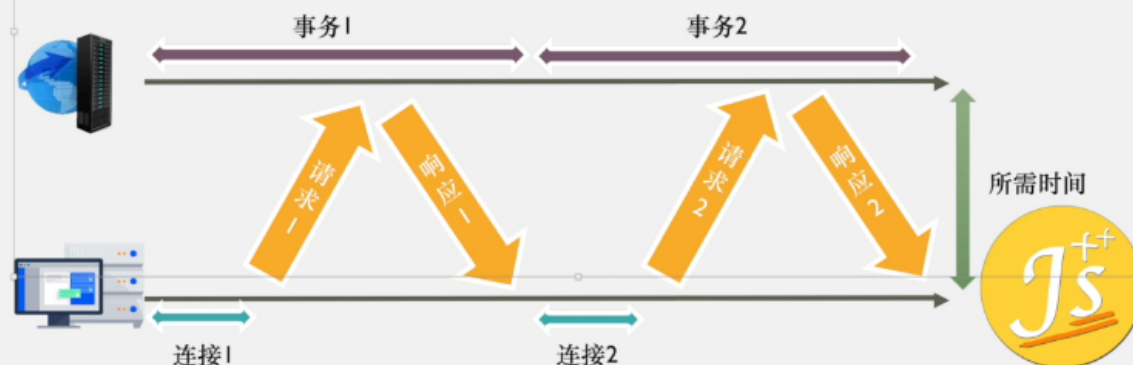
- GET/POST请求方式

- 4、POST比GET速度慢

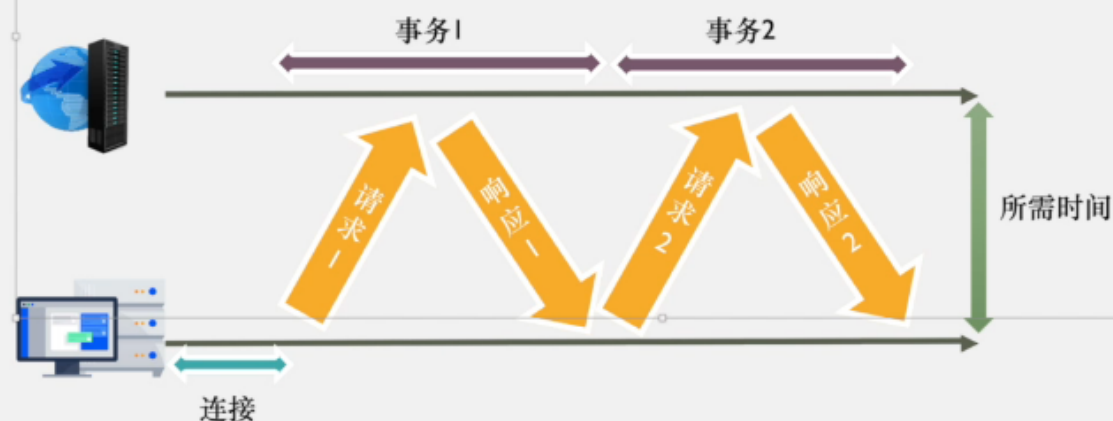
- 3、GET会进行数据缓存，POST不会

The screenshot displays the network activity in a web browser. The 'Headers' tab for a GET request to 'http://img.jsplusplus.com/website/teacher/1.jpg' is active. Key details include the request method 'GET', status '200 OK (from memory cache)', and various response headers like 'Accept-Ranges: bytes' and 'Access-Control-Allow-Origin: *'. The 'Timing' tab on the right lists multiple resources, with '1.jpg' specifically highlighted in red and noted as being served from the memory cache.

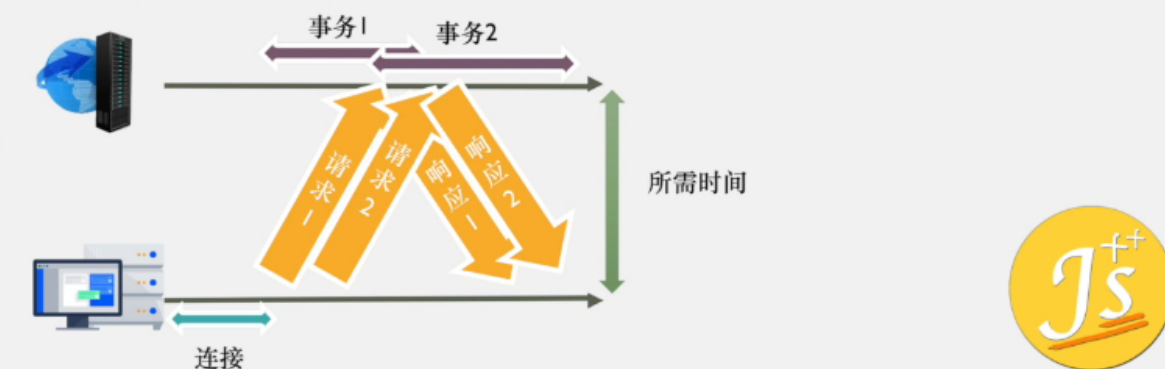
- GET/POST请求方式
- 4、POST比GET速度慢
- 4、POST不能进行管道化传输
- 1、串行连接



- GET/POST请求方式
- 4、POST比GET速度慢
- 4、POST不能进行管道化传输
- 1、持久化连接 (connection: keep-alive) : (HTTP/1.0、1.1) 连接不会关闭



- GET/POST请求方式
- 4、POST比GET速度慢
- 4、POST不能进行管道化传输
- 1、管道化持久连接 (http/1.1-> 把所有请求放到发送队列里, 不等响应, 一个一个发送请求的同时接收相应的响应)



- 管道化持久连接的弊端：一旦TCP/IP连接中断，只能重新提交队列，这时如果存在支付业务，就要在支付一次


HTTP报文-请求方式

- GET/POST请求方式
- 数学中的幂等： $x^y = x$ $x=0 || 1$
- 幂等性：一个HTTP请求中，不会对数据或状态做修改，并且每次请求都会返回同样的结果。
- 必要性：1、GET请求必须遵守幂等性，从HTTP请求上来看，GET只能获取数据。
- 2、POST请求一般做增删改的操作，所以一般不遵守幂等性
- 管道化传输不可以用非等幂性请求的原因
- 假设：一个管道中由10个请求，发送了9个，但是连接关闭了，即使收到了9个响应，这9个响应内容也将被清空，那么客户端将会重新发起这9个请求，但是9个响应收到就证明服务器已经做了相应的操作，如果是非幂等的请求，则会出现操作的错误（如支付、增删改数据等）
- 所以管道化传输不支持非幂等的请求，POST是幂等或非幂等请求，都不支持管道化传输。
- 幂等性重点：每次请求都返回相同的结果
- POST请求不具有幂等性，所以不能使用管道化传输
- 管道化持久连接不支持POST方式
- 现在浏览器大多数使用Keep-alive方式，要使用管道化连接需要单独配置

HTTP报文状态码

1441115231488510110正在观看视频

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



- 304 重定向
- Etag: 服务端资源唯一标识符 (优先级高于Last Modified)
- Last-modified: 资源在服务器最后修改的时间 (精确到秒) -> 所以需要唯一标识符
- 1、第一次访问index.html (响应头) - 200 OK

Date: Sat, 05 Jan 2019 13:55:46 GMT

ETag: "144-57eb65254f380"

Keep-Alive: timeout=5, max=100

Last-Modified: Sat, 05 Jan 2019 13:53:34 GMT



- 客户端
 - if-Modified-Since: Sat, 05 Jan 2019 13:53:34 GMT
 - if-None-Match: "144-573b65254f380"
 - 存储了第一次访问index.html 时的服务端返回头的ETag和Last-Modified 的值
 - 当再次请求时, 请求头携带这两个属性给到服务端, 服务端用ETag 和 Last-Modified 的值和客户端传来的这两个值进行比对, 如果都相同, 则告诉客户端从缓存获取index.html 这个文件, 这时传来的状态码就是304
 - Last-Modified: 使用的是GMT 也叫 世界协调时, 中国是东八区 (东加西减), 所以中国的时间是在GMT时间上加8

- 304 重定向

- 2、第二次访问index.html (请求头) 304 Not Modified

If-Modified-Since: Sat, 05 Jan 2019 13:53:34 GMT

If-None-Match: "144-57eb65254f380"



- 304 重定向

- 3、修改index.html第三次访问 (响应头) 200 OK

ETag: "148-57eb661696cc0"

Keep-Alive: timeout=5, max=100

Last-Modified: Sat, 05 Jan 2019 13:57:47 GMT

If-Modified-Since: Sat, 05 Jan 2019 13:53:34 GMT

If-None-Match: "144-57eb65254f380"



144115

- 302 重定向

- 1、服务端程序重定向（跳转<demo.html>）返回302 Found

- 404 页面错误

- <http://localhost/network/class5/index2.html>（页面不存在）

- 403 服务器拒绝请求 forbidden



- 500 Internal Server Error 服务器发生不可预测的错误



- 503 server Unavailable 服务器当前不能处理客户端请求(关闭应用程序池或者程序标识出错或者程序池队列已满)



- 503 可能只是临时错误，比如服务端程序池已满，可能下一秒就可以访问了

HTTP 报文- ACCEPT & Content-Type

HTTP报文-ACCEPT & CONTENT-TYPE

- Accept: 代表客户端希望接收的数据类型

▼ Request Headers view source

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cache-Control: max-age=0
Connection: keep-alive
Cookie: BDUSS=E2UHNuQmN1SVZwMfN0VG1tYn5mUHZPN1pahk8zZwPwMwVDM3hqem8zZ1AtUmhjQVFBQUFBjCQAAAAAAAAAAAAAAACf6kaCY29kZXJfbnVtMQAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA9s8VsPbPfBmU; BAIDUID=4CBE2AFD48915BCCF29F5D01AFDABA38; FG=1; PSTM=1546407317; BIDUPSID=55F71B08ED224F5F198B3117
6E69FC98; BD_UPN=123253; BDORZ=FFFB88E999055A3F8A630C64834B0D6D0; delPer=0; BD_CK_SAM=1; PSIN0=3; pgv_pvi=1343364096; pgv_si=s8781010944; H_PS_645EC=c544
6d2XBdcjcoY9%2B%2B5N53QP9wBjK2cRzh3fXwpqGvgeWLEsFByV3DQ5Kd%2FHljyth5TDHXg; BD_HOME=1; H_PS_PSSID=1430_21118_20880_28206_28132_26350_28139_22075
Host: www.baidu.com
Referer: https://www.baidu.com/s?ie=utf-8&f=8&rsrv_bp=1&tn=06074089_54_pg&wd=http%20expres&oq=http%2520content%2520type&rsrv_pq=d951790900019070&rsrv_t=9bd2
L0VyQITxcxcF58AwwT04I3gPW7u4WfMaDqFYcIT9G7cJKNORRwyJhHoNQ0cu%2FT14&rlang=cn&rsrv_enter=0&inputT=6411&rsrv_sug3=97&rsrv_sug1=48&rsrv_sug7=100&bs=http%20c
ontent%20type
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36
```

- q: 相对品质因子, 权重, 它从0到1的范围指定优先顺序, 没有指定, 质量值默认为“q = 1”,
- 如赋值为0, 则提醒服务器该内容类型不被浏览器接受
- **Accept:** (请求头) `text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8`
- type; q=value, type; q=value
- 最希望接收到`text/html,application/xhtml+xml`, 其次是`application/xml`, 再其次是其他任意数据类型
- **Content-Type** (响应头)
- **Content-Type:** `text/html; charset=UTF-8` (Accept-Charset)
- 返回的资源类型与编码
- *: 通配符, /*任意类型

- Accept 请求头中, 是, 号分割

- **Accept-Language**
- **Accept-Language:** `zh-CN,en-US;q=0.8,en;q=0.6`
- 浏览器支持的语言是简体中文、其次是美国英语、再其次是其他形式的英语
- **Content-Language**
- **Content-Language:** `zh-CN`
- 说明返回资源的语言类型



• Accept-Encoding

• **Accept-Encoding:** gzip, deflate, br

• 浏览器可以接受的资源编码格式（压缩格式）

• Content-Encoding

• Content-Encoding: gzip

• 服务器返回资源的编码格式（压缩格式，优化传输内容的大小）

- 压缩格式： 优化传输内容的大小