

Http 面试总结

目录:

- [1、常用的 HTTP 请求方法有哪些？](#)
- [2、GET 方法与 POST 方法的区别](#)
- [3、HTTP 请求报文与响应报文格式](#)
- [4、常见的 HTTP 相应状态码和请求头字段](#)
- [5、HTTP1.0 HTTP 1.1 HTTP 2.0 主要区别](#)
- [6、断点续传，文件更新后如何续传？](#)
- [7、常见 HTTP 首部字段有哪些？](#)
- [8.HTTP 的缺点与 HTTPS 的区别，HTTPS 的工作原理？](#)
- [9、HTTP 优化](#)
- [10 当输入 www.google.com 时，页面发生了哪些事情：](#)
- [11. 强缓存和协商缓存的命中和管理](#)
- [12 请列举三种禁止浏览器缓存的头字段，并写出相应的设置值](#)
- [13 cookie 和 session 的区别：](#)
- [14 cookie 由哪几部分组成？](#)
- [15 DNS 递归解析和迭代解析](#)

1、常用的 HTTP 请求方法有哪些？

答：

GET: 用于请求访问已经被 URI（统一资源标识符）识别的资源，可以通过 URL 传参给服务器

POST: 用于传输信息给服务器，主要功能与 GET 方法类似，但一般推荐使用 POST 方式。

PUT: 传输文件，报文主体中包含文件内容，保存到对应 URI 位置。

HEAD: 获得报文首部，与 GET 方法类似，只是不返回报文主体，一般用于验证 URI 是否有效。

DELETE: 删除文件，与 PUT 方法相反，删除对应 URI 位置的文件。

OPTIONS: 查询相应 URI 支持的 HTTP 方法。

2、GET 方法与 POST 方法的区别

区别一：

get 重点在从服务器上获取资源，post 重点在向服务器发送数据；

区别二：

get 传输数据是通过 URL 请求，以 field（字段）= value 的形式，置于 URL 后，并用 "?" 连接，多个请求数据间用 "&" 连接，如 <http://127.0.0.1/Test/login.action?name=admin&password=admin>，这个过程用户是可见的；

post 传输数据通过 Http 的 post 机制，将字段与对应值封存在请求实体中发送给服务器，这个过程对用户是不可见的；

区别三：

Get 传输的数据量小，因为受 URL 长度限制，但效率较高；

Post 可以传输大量数据，所以上传文件时只能用 Post 方式；

区别四：

get 是不安全的，因为 URL 是可见的，可能会泄露私密信息，如密码等；

post 较 get 安全性较高；

区别五：

get 方式只能支持 ASCII 字符，向服务器传的中文字符可能会乱码。

post 支持标准字符集，可以正确传递中文字符。

3、HTTP 请求报文与响应报文格式

请求报文包含三部分：

- a 请求行：method 标识+request URI + version
- b 请求报头：不同的报头域描述不同的请求参数，如 Accept 、Accept-Encoding.....
- c 请求正文

响应报文包含三部分：

- a 状态行：版本号 状态码 状态码描述文本
- b 响应报头：如 Location、 Server



4、常见的 HTTP 相应状态码和请求头字段

200: 请求被正常处理

204: 请求被受理但没有资源可以返回

206: 客户端只是请求资源的一部分，服务器只对请求的部分资源执行 GET 方法，相应报文中通过 Content-Range 指定范围的资源。

301: 永久性重定向

302: 临时重定向

303: 与 302 状态码有相似功能，只是它希望客户端在请求一个 URI 的时候，能通过 GET 方法重定向到另一个 URI 上

304: 资源如果没有改变就从缓存中获取

307: 临时重定向，与 302 类似，只是强制要求使用 POST 方法

400: 请求报文语法有误，服务器无法识别

401: 请求需要认证

403: 请求的对应资源禁止被访问

404: 服务器无法找到对应资源

500: 服务器内部错误

503: 服务器正忙

请求头字段:

消息头字段: cache-control, Date, Connection(none/keep-alive)

请求头字段: Accept, Host, If-Range, If-Modified-Since, Range, User-Agent

响应头字段: last-modified, location, ETag, Age....

实体头字段: Content-*, Expire

5、HTTP1.0 HTTP 1.1 HTTP 2.0 主要区别

HTTP1.0 HTTP 1.1 主要区别

(1) **长连接:** http 1.0 需要 keep-alive 参数来告知服务器端需要建立一个长连接, 而 http1.1 默认支持长连接

(2) **节约带宽:** http1.1 支持只发送 header 信息, 不带 body, 如果客户端有权访问, 服务器就返回 100, 否则返回 401, 客户如果接受到 100, 就发送 body, 否则就不发送 body, 节约带宽

(3) **host 域:** http1.1 在请求消息的报文头中加入了 host 域

(4) **ETag 头** : http1.1 中引入了 ETag 头, 它的值 entity tag 可以用来唯一的描述一个资源. 请求消息中可以使用 If-None-Match 头域来匹配资源的 entitytag 是否有变化

(5) **Cache-Control** : http1.1 新增了 Cache-Control 头域(消息请求和响应请求都可以使用), 它支持一个可扩展的指令子集

(6) **Warning 头域:** http1.0 中只定义了 16 个状态响应码, 对错误或警告的提示不够具体. http1.1 引入了一个 Warning 头域, 增加对错误或警告信息的描述. 且新增了 24 个状态响应码

HTTP 2.0 主要改进

(1) **多路复用:** 一个连接可以发送多个请求和响应, 每个请求或者响应可以看作是一个流, 每个流是并发或者并行处理的, 互不影响,

而 http1.1 采用的流水线处理方式，虽然一个连接可以发送多个请求或者响应，但是后面的流必须等待前面的流，容易阻塞

(2) 二进制格式：错误少，效率高

(3) 报头压缩：有的消息头很大，由于 tcp 是慢启动，需要几个来回才能确定消息的参数信息和包的数量，压缩报头能够节省很多时间

(4) 服务器将响应主动推送到客户端的缓存中：比如浏览一个网页时，服务器返回 html，服务器发送 js，css，图片等资源前需要等待浏览器解析 html 并响应，如果服务器主动推送这些静态数据，将提高效率。

6、断点续传，文件更新后如何续传？

http1.1 支持在请求报文头中加入 range 字段和 content-range 字段，如：

- 1 客户端下载一个 1024K 的文件，已经下载了其中 512K
- 2 网络中断时，就将请求报文头 range=512k
- 3 服务器接受后将 Content-range 设为 512k-1024k 返回状态码 206 而不是 200

如果文件更新，续传的文件就不是源文件了，这时候就应该加入一个标识判断文件是否发生变化，如 last-modified 标识最后修改时间 Etag 标识文件的 MD5 值等，客户端发送请求时可以发送一个 if-range 头，存放原来文件的 ETag 或者 last-modified，服务器通过校验，不一致则返回 200，和新文件，一致时返回 206 继续续传

7、常见 HTTP 首部字段有哪些？

a、通用首部字段（请求报文与响应报文都会使用的首部字段）

Date: 创建报文时间

Connection: 连接的管理

Cache-Control: 缓存的控制

Transfer-Encoding: 报文主体的传输编码方式

b、请求首部字段（请求报文会使用的首部字段）

Host: 请求资源所在服务器

Accept: 可处理的媒体类型

Accept-Charset: 可接收的字符集

Accept-Encoding: 可接受的内容编码

Accept-Language: 可接受的自然语言

c、响应首部字段（响应报文会使用的首部字段）

Accept-Ranges: 可接受的字节范围

Location: 令客户端重新定向到的 URI

Server: HTTP 服务器的安装信息

d、实体首部字段（请求报文与响应报文的的实体部分使用的首部字段）

Allow: 资源可支持的 HTTP 方法

Content-Type: 实体主类的类型

Content-Encoding: 实体主体适用的编码方式

Content-Language: 实体主体的自然语言

Content-Length: 实体主体的字节数

Content-Range: 实体主体的位置范围，一般用于发出部分请求时使用

8. HTTP 的缺点与 HTTPS 的区别，HTTPS 的工作原理？

- a、通信使用明文不加密，内容可能被窃听
- b、不验证通信方身份，可能遭到伪装
- c、无法验证报文完整性，可能被篡改

HTTPS 就是 HTTP 加上加密处理（一般是 SSL 安全通信线路）+认证+完整性保护

1. HTTP 的 URL 以 http:// 开头，而 HTTPS 的 URL 以 https:// 开头
 2. HTTP 是不安全的，而 HTTPS 是安全的
 3. HTTP 标准端口是 80 ，而 HTTPS 的标准端口是 443
 4. 在 OSI 网络模型中，HTTP 工作于应用层，而 HTTPS 的安全传输机制工作在传输层
 5. HTTP 无法加密，而 HTTPS 对传输的数据进行加密
 6. HTTP 无需证书，而 HTTPS 需要 CA 机构 wosign 的颁发的 SSL 证书
- HTTPS 工作原理：HTTP + SSL（Secure Socket layer）/TLS（transfer layer security）

- 1 客户端发起 https 请求,连接到服务器的 443 端口，
- 2 服务器配置一套数字证书,可以自己生成(需要客户端验证)，也可以向信任的公司申请，这套证书就是公钥和密钥，

- 3 服务端发送公钥（包括证书的机构，过期时间）
- 4 客户端解析证书（客户端的 TLS 完成的），如果正确就是生成一个随机值，然后用证书对随机值加密
- 5 客户端把加密后的随机值发送到服务器，以后用这个随机值进行通信加密
- 6 服务端解密随机值，然后用随机值对内容进行对称加密（将信息和密钥混在了一起），这样除非知道密钥，否则无法获取内容
- 7 服务端传输加密后的数据
- 8 客户端用随机值解密

9、HTTP 优化

利用负载均衡优化和加速 HTTP 应用

利用 HTTP Cache 来优化网站

10 当输入 `www.google.com` 时，页面发生了哪些事情：

- 1.域名解析域名解析检查顺序为：浏览器自身 DNS 缓存---》OS 自身的 DNS 缓存--》读取 host 文件--》本地域名服务器--》权限域名服务器--》根域名服务器。如果有且没有过期，则结束本次域名解析。域名解析成功之后，进行后续操作
- 2.tcp3 次握手建立连接
- 3.建立连接后，发起 http 请求

- 4.服务器端响应 http 请求，浏览器得到到 http 请求的内容；
- 5.浏览器解析 html 代码，并请求 html 代码中的资源
- 6.浏览器对页面进行渲染，展现在用户面前。

11. 强缓存和协商缓存的命中和管理

- 1) 浏览器在加载资源时，先根据这个资源的一些 http header 判断它是否命中强缓存，强缓存如果命中，浏览器直接从自己的缓存中读取资源，不会发请求到服务器。
- 2) 当强缓存没有命中的时候，浏览器一定会发送一个请求到服务器，通过服务器端依据资源的另外一些 http header 验证这个资源是否命中协商缓存，如果协商缓存命中，服务器会将这个请求返回，但是不会返回这个资源的数据，而是告诉客户端可以直接从缓存中加载这个资源，于是浏览器就又会从自己的缓存中去加载这个资源；
- 3) 强缓存与协商缓存的共同点是：如果命中，都是从客户端缓存中加载资源，而不是从服务器加载资源数据；区别是：强缓存不发请求到服务器，协商缓存会发请求到服务器。
- 4) 当协商缓存也没有命中的时候，浏览器直接从服务器加载资源数据。

(当 **ctrl+f5** 强制刷新网页时，直接从服务器加载，跳过强缓存和协商缓存；

当 **f5** 刷新网页时，跳过强缓存，但是会检查协商缓存；)

12 请列举三种禁止浏览器缓存的头字段，并写出相应的设置值

Expires: 告诉浏览器把回送的资源缓存多长时间 -1 或 0 则是不缓存
Cache-Control: no-cache
Pragma: no-cache

13 cookie 和 session 的区别：

1cookie 数据存放在客户的浏览器上，session 数据放在服务器上。

cookie 不是很安全，别人可以分析存放在本地的 COOKIE 并进行 COOKIE 欺骗

考虑到安全应当使用 session。

2 session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能 考虑到减轻服务器性能方面，应当使用 COOKIE。

3 单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie。

所以建议是：

将登陆信息等重要信息存放为 SESSION

其他信息如果需要保留，可以放在 COOKIE 中

14 cookie 由哪几部分组成？

Set-Cookie: NAME=VALUE ; Expires=DATE ; Path=PATH ;

Domain=DOMAIN_NAME; SECURE

15 DNS 递归解析和迭代解析

1 递归解析：客户端（浏览器）查询自身 DNS 缓存->OS 自身缓存->host 文件->本地域名服务器->

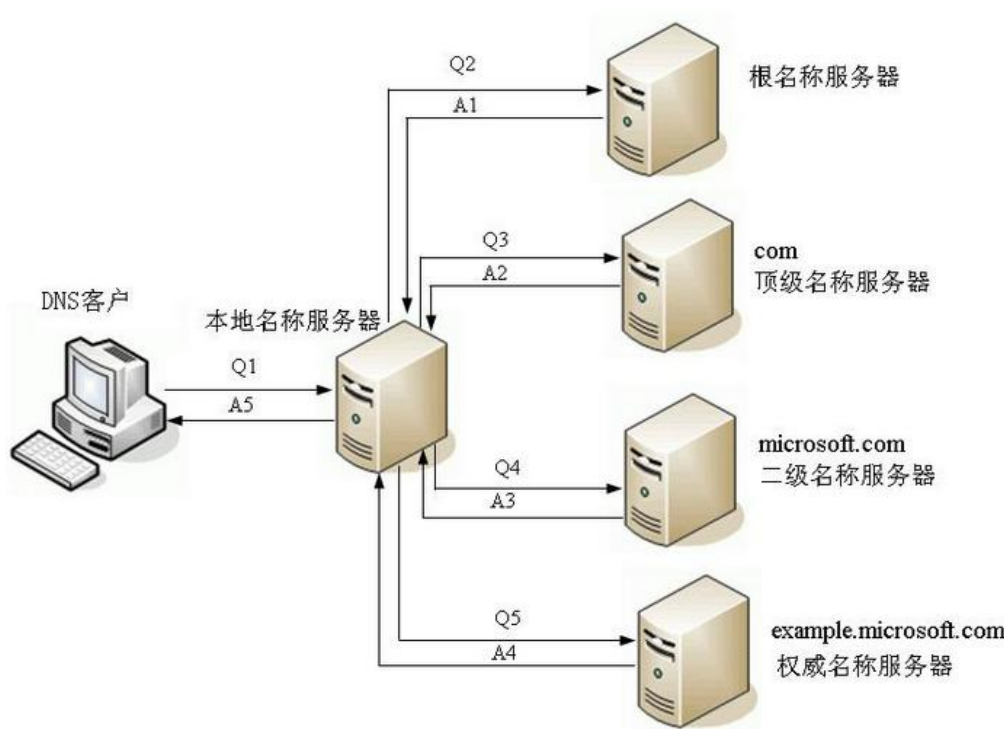


图11-15 DNS递归名称解析示例

2 迭代解析：客户端（浏览器）查询自身 DNS 缓存->OS 自身缓存
->host 文件->本地域名服务器->

