OSI 七层模型

- 应用层：网络应用协议和网络协议 HTTP 存在的一层。比如浏览器和 http/HTTPS/SSH/FTP 协议，DNS协议等
- 表示层：数据压缩和数据加密的
- 会话层：建立、管理、终止会话
- 传输层：传输应用程序报文。这一层存在两个重要的协议 TCP/UDP协议
- 网络层：IP层，RIP, OSPF。
- 链路层：ARP
- 物理层

> SSL/TLS协议位于应用层和传输层之间，数据包加密

> 应-》表-》会-》传-》网-》链-》物

应：HTTP/HTTPS/FTP/DNS/SSH 传: TCP/UDP 网: IP，RIP, OSPF 链：ARP

CDN

Content Delivery Network，即内容分发网络，CDN 利用了 HTTP 的缓存和代理技术，在现有网络的基础上将源站点的服务器内容分发，分发到部署的所有服务器上，这样用户就可就近获取资源，提高了访问的速度，和降低了源站点服务器的压力。

- CDN 的主要功能是做内容存储和分发技术的。

CDN 原理

- 当用户 访问站点资源时，首先需要我们的 DNS 解析的 IP 地址，如果 DNS 有缓存的 IP 地址记录，CDN 的全局负载均衡设备 就会根据得到的 IP 地址，选择距离用户最近的节点发起请求。如果没有 IP 地址的解析记录，则需要 DNS 解析得到 IP 地址，再交给 CDN 的全局负载均衡设备发起请求。
  - 如果该 IP 地址对应的节点(服务器)有缓存的资源。那么直接将资源放回给用户
  - 如果该 IP 地址对应的节点没有缓存到资源，那么就需要向源站点服务器获取资源，结合缓存策略，缓存到节点上，再返回给用户

CDN 优点

- 访问速度快。
- 获取资源的速度快
- 减少了服务器的压力，提高了并发的能力

HTTP 的报文结构

> HTTP 的报文结构分成请求报文，响应报文两种

HTTP 请求报文结构

> HTTP 报文结构：请求行，空行，请求头，请求体

- 请求行是，请求方法+路径+HTTP版本

```
请求报文：GET /home HTTP/1.1
响应报文：HTTP1.1 200 ok
```

- 请求头部 请求头

> origin，content-type：json/，accept-language，cache-control: no-cache。Connection，cookie，Host

```
User-Agent：产生请求的浏览器类型。
Accept：客户端可识别的内容类型列表。
Host：请求的主机名，允许多个域名同处一个IP地址，即虚拟主机。
```

- 请求体，请求体携带 post，put，patch 等请求携带的数据，推送到服务器端

HTTP 响应行报文

> HTTP 的响应报文分成 响应行，空行，响应头，响应体

- 响应行，由 响应状态码的原因短语，状态码，HTTP版本号 组成

```
HTTP1.1 200 ok
```

- 响应头，响应回来的头部字段信息

```
// 允许跨域请求的域名
access-control-allow-origin: https://juejin.cn
// HTTP 的一种加密算法
content-encoding: br
// 响应的文件类型
content-type: application/json; charset=utf-8
```

- 响应体，服务器响应的内容

HTTP 请求的 9 种方法

- get 一般用于获取资源
- head 用于获取请求头的资源
- post 用于发送资源
- put：一般用于修改资源
- delete：删除某一些资源
- CONNECT：建立连接，一帮用于代理服务器
- option：列出资源的请求方式有哪些，用于跨域请求
- trace：用于追踪请求和响应的传输路径。
- PATCH: 用于对资源进行部分修改

get, delete, head, post, put, connection, trace, patch, option

POST 和 GET 有哪些区别

> http 的所有请求方法中都可以从服务端获取数据，和传递内容。get：主要是从服务器获取数据。post 主要发送数据给服务器。GET 和 POST 本质上就是 TCP 链接，并无差别，但是由于 HTTP 的规定和浏览器/服务器的限制具体有如下的区别

- 从缓存的角度上说，get 请求会被浏览器默认缓存下来，而 post 请求默认不会。
- 从参数来说，get 请求的参数一般放在 url 中，post 请求是放在请求主体 body 中，因此post 请求更安全一些。
- 从 TCP 上来说，GET 产生一个 TCP 数据包；POST 产生两个 TCP 数据包。对于GET方式的请求，浏览器会把 请求头，请求体 http header 和 data 一并发送出去，服务器响应200（返回数据）；而对于POST，浏览器先发送 header，服务器响应 100 continue，浏览器再发送 data，服务器响应 200 ok（返回数据）。虽然 post 请求需要发送两次，但是时间上是基本差别不大的。还有并不是所有浏览器都会在 POST 中发送两次包，Firefox 就只发送一次
- PUT 幂等（幂等的概念是指同一个请求方法执行多次和仅执行一次的效果完全相同）用于修改数据库中的某一个字段，而 POST 是非幂等。用于向服务器中新增一个字段内容

HTTP 状态码

- 1xx，表示目前还在处理协议的中间状态，还需要后续的操作
- 2xx，表示处理成功状态
- 3xx，表示重定向，资源位置已经被移动，需要重新请求
- 4xx，表示客户端请求的报文有误，不被服务端理解
- 5xx，服务端发生错误

> 1: 表示请求还没有处理完成，2：开头的表示请求处理成功，3.开头的表示请求的资源位置发生了改变，4，开头的表示请求的资源不被服务器理解。5开头表示服务器发生错误。

-
  - 2xx：表示处理成功状态。常见的有， 200：表示 ok，表示服务能够返回信息。 204：No Content 无内容，没有响应数据回来。服务器成功处理，但未返回内容。 206：Partial Content 表示部分内容，使用场景是 HTTP 分块下载和断点续传。

一类重要的高频考点

-
  - 3xx：表示重定向，资源位置已经被移动，需要重新请求。 301：表示请求的资源被永久转移，返回旧域名会跳转到新域名。 302：临时转移。表示请求的资源临时被转移了。由于历史原因，302 重定向后可能会把 POST 请求改为 GET请求 304：表示不设置缓存，已经有协商缓存，对于不经常更新的文件，例如 css/js/html文件，服务器会结合客户端设置 304 状态。加载过的资源下次请求时会在客户端中获取。 307：表示临时重定向。为了弥补 302 的缺陷。 **302和307状态码表示的含义一样，唯一的区别在于 307 不允许将请求方法从 post 改为 get。**
-
  - 4xx：表示客服端请求出错，请求无法被服务器端理解。 400：表示请求的参数错误，没有具体的错误信息 403 forbidden ：表示无权限访问。 404：表示请求的资源没有找到。 413：表示和服务器的交互过大。
-
  - 5xx：服务器端出错。 500：表示服务器端出现未知的错误。 503：服务器超负荷。

HTTP 特点和缺点(应用层)

- 特点：
  - 灵活，对格式上没有过多的要求，可以存在多种数据传输的格式，可以是文本，图片，视频等
  - 可靠，HTTP 是基于 TCP/IP 实现的。具备 TCP 的特性
  - 有请求就有应答
  - 无状态，没有记录每一次 HTTP 请求，每一次请求都是独立无关的。
- HTTP 缺点：
  - 无状态，这是特点也是缺点，HTTP 不记录用户的操作，比如用户登陆网址后输入密码和账号，第二次登陆时需要重新登陆，因为 HTTP 没有记录用户的操作。所以这是 cookie 引入的原因。
  - 明文传输（不安全），传输的报文是文本协议，不是二进制格式(HTTP2引入二进制传输)。明文传输容易被攻击。

> HTTP 特点：灵活，支持多种文件传输格式。可靠，基于TCP/IP实现。无状态，请求的HTTP不会被记录下来。 HTTP 缺点：无状态，没有办法记录上一次的登陆状态。每一次都需要登陆，所以诞生了 cookie，来解决 HTTP 的无状态。明文传输：HTTP的数据传输时明文格式的不安全，所以在 HTTP2.0时引入了二进制传输。

Access 系列字段

- 数据格式：Content-Type：字段是 HTTP 支持的数据类型，包括文本，图片，音频和媒体，json，JavaScript 等等
- 压缩方式：content-encoding：内容压缩的方式，gzip 最流行，deflate，br 压缩算放

```
// 发送端
Content-Encoding: gzip
// 接收端
Accept-Encoding: gzip
```

Cookie

- cookie 由服务端生成。cookie 就是一个存放在客户端的一个小文件，也可以存放在本地，假设浏览器关闭后 cookie 依旧存在 https://juejin.cn/post/6844904095711494151#heading-1

cookie 的设置过程

1. 客户端向服务器发送一个 HTTP 请求。
2. 服务器接收到请求后在响应头添加一个 `set-cookie` 的字段
3. 客户端接收到服务器的响应后将 cookie 保存下来，保存到本地的文件夹内
4. 之后浏览器每一次请求都会携带 cookie 发送给服务器（耗性能）

- Cookie 的某一些属性
  - Name：就是 cookie 的属性名
  - Value：cookie 的属性值，需要做编码处理

    - Expire：设置 cookie 的过期时间，Set-Cookie:Expires=Wed, 21 Oct 2015 07:28:00 GMT; **需要注意的是如果 cookie 的 Expire 没有设置，那么表示这个 cookie 是会话 cookie 浏览器关闭后就消失了。持久性的 cookie 是存放在硬盘中的，直到时间过期或者手动清除**
    - max-age：表示 cookie 失效之前的秒数，`Set-Cookie:Max-Age=604800;` 这个属性的值可以是正数：表示持久性的 cookie。负数：表示会话的 cookie 浏览器关闭就消失。0：表示立即删除这个cookie。**当 Max-Age 和 Expire 都存在是，Max-Age 的优先级更高**
    - size：cookie 的大小，超过 4kB 后会被忽略。
    - Domain: 记录域名信息， 但是不能跨站点设置域名，不会起作用
    - path：指定的要发送的 URL 路径。
    - SameSite：限制第三方对 cookie 的携带请求，这属性可以防止 CSRF 攻击。三个重要的属性 strict：仅仅允许同一个域名的站点携带 cookie。Lax：允许第三方携带 cookie 值。None：表示不允许任何站点携带 cookie。一开始默认值是 None，后来默认值是 Lax。
    - HttpOnly: 限定 cookie 只能通过 HTTP 传输，JavaScript 不能读取，防止 XSS 攻击
    - Secure：限定了只有 HTTPS 才可以传输 cookie

- 优点：

    - 记录用户的操作状态，密码等
    - 弥补了 HTTP 的无状态

- 缺点：

    - 容量小只有 4kb
    - 不安全，可以随意修改 cookie 内容
    - 耗费性能，每一次请求都会携带完整的 cookie

> domain：域名信息， path：cookie的路径，sameSite：限制第三方对 cookie 的读取，有效防止 CSRF 攻击。HttpOnly：限制 JavaScript 对 cookie 的读取，有效防止 XSS 的攻击。secure：限制了只有 HTTP 才可以读取 cookie。优点：有状态能记录用户的操作行为账号密码等。弥补 HTTP 的无状态。缺点：体积小只有 4kb，而且不安全cookie可以被修改。消耗性能，每一次请求都会携带 cookie。

HTTP 的三种缓存 默认端口号80

- 强缓存
- 协商缓存
- 代理缓存：所谓的代理缓存是指，让代理服务器分担一部分源服务器的 HTTP 缓存，客户端的缓存过期后就从代理缓存中获取，如果代理服务器的缓存也过期了，再从源服务器中获取资源没降低服务器的压力

怎么实现 HTTP 的代理缓存

> HTTP 的代理缓存主要分为两部分实现，一部分是源服务器的缓存控制，一部分是客户端的缓存设置。

- 源服务器的缓存控制 catch-control

    - catch-control 可以来控制代理服务器能否设置的缓存。
    - 源服务器的响应头中可以添加 catch-control: private 或 public。
    - private：表示代理服务器不可以设置缓存，public 表示可以设置缓存内容
    - proxy-revalidate：代理缓存过期后就从服务器中获取
    - s-maxage：2000：表示代理缓存在代理服务器中可以存放多久，这里是2000s

```
Cache-Control: public, max-age=1000, s-maxage=2000
```

- 客户端缓存实现控制，客服端的缓存控制是在请求头中的
    - 客服端的缓存控制是对代理服务器的限定
    - 请求头中添加 max-stale:5 或 min-fresh:5
    - only-if-catch: 这个字段表示，客服端只接受代理服务器的缓存，不接受服务器的缓存，如果代理服务器的缓存无效则，服务器则直接放回 504

参考http缓存

HTTPS (HTTPS = HTTP + SSL/TLS) 默认端口号 443

https://wetest.qq.com/lab/view/110.html

> HTTPS 并不算是一个新的协议，只是在 HTTP 的基础上添加了 SSL/TLS 协议

- HTTP 是明文传输的，对于一些敏感信息的传输就显得不太安全，HTTPS 就是为了解决这个问题出现了，是加密传输的

HTTPS 是怎样保证数据传输的安全性的/https 的加密算法？

> HTTPS 能保证数据安全的传输实现有两种方式，对称加密 和 非对称加密，这两种加密算法都是可逆的

- 对称加密算法：所谓的对称加密是通信双方通过密钥来加解密
    - 优点：简单和性能好
    - 缺点：无法解决 首次发送 把密钥发送给对方的的问题，容易被拦截
    - 常用的对称加密算法有：DES, 3DES, AES,RC4, RC5, RC6

> 扩展：DES 对称加密算法，DES 是目前最流行的加密算法之一，对称加密算法的一种，而且还是分组算法，以64位为一组。其中有 56 位为加密密钥，密钥通常是 64位，但是每一个第8位适用于奇偶校验的，所以就剩下了56位的加密密钥。

- 非对称加密：私钥+公钥 = 密钥对，用私钥加密的数据需要用公钥来解开。用公钥加密的数据需要使用私密解开。在通信之前双方会都将自己的公钥向发送给对方，等接收到对方的数据后再拿接收到的公钥来加密数据，最后响应给对方，对方接收到数据后使用私有的私钥解密即可。
    - 优点：安全性更高
    - 缺点：速度慢，影响性能
    - 常用的非对称加密算法：RSA，
- 解决方法：结合使用对称加密和非对称加密，混合加密。将对称加密的密钥使用非对称加密的公钥加密，发送到对方后，对方使用私钥解密得到对称加密的密钥，之后双方通过对称加密的密钥进行通信。

> 混合加密：使用非对称加密的公钥加密对称加密的密钥，然后发送给对方，对方接收到以后，使用私钥解密得到对称加密的密钥，之后使用对称加密的方式通信。其实这样就是解决了对称加密首次发送给对方容易被劫持的缺点

- CA 证书 **数字签名，是第三方办理的证书，这个证书包含了签发者的个人信息，使用者的密钥，私钥等，证书会使用 hash 算法得到内容的一个摘要。再用第三方证书的密钥加密，最终得到数字签名。** CA 机构拥有非对称加密的私钥和公钥。
    - 数字签名常用的加密算法：MD5，MAC。

> MD5 信息摘要算法，是不可逆的加密算法之一，会生成 128 位的 hash 值，利用的就是哈希函数。

SSL 的五次握手过程

- 1. 客户端发出一个 HTTPS 的请求，同时发送生成随机数和客户端支持的加密算法
- 2. 服务器接收到请求后，会发送数字证书和自己生成的随机数。
- 3. 客户端接收到这份证书后，再生成一个新的随机数，使用证书中的公钥(非对称加密的公钥)，加密这个随机数
- 4. 服务端接收到后使用自己的私钥解密公钥得到这个随机数。
- 5. 最后使用前面三个随机数，结合支持的算法，生成会话密钥(对称密钥)。这个会话密钥用于后面的会话。 阮一峰

https 的缺点

HTTP1.0 和 HTTP1.1 的区别

- 缓存处理改变。强缓存和协商缓存标识有发生变化，`HTTP1.0` 中使用 `Expires强缓存和 Last-Modify 协商缓存`，`HTTP1.1 中使用, Cache-Control 强缓存 /Etag协商缓存`。
- 增加长连接 `Connection`: keep-alive。弥补了 HTTP1.0 每次请求都需要连接的缺点。
- `HTTP1.1 新增了五种请求方法`：OPTIONS, PUT, DELETE, TRACE 和 CONNECT
- 宽带优化和网络连接的使用，HTTP1.1 支持断点续传，返回206码。
- 新增错误状态码，HTTP1.1 新增了 24 个错误的状态码，比如409，表示请求的资源和当前资源发生冲突，410：标识请求的资源在服务器上已经被永久的删除。
- `Host` 头处理改变。在 `HTTP1.0` 中默认一台主机只有一个 IP 地址。因此请求的 `URL` 中并没有主机名 hostname。但是虚拟主机的出现，一台服务器可以存在多个虚拟主机，虚拟主机之间可以共享一个 IP 地址。在 HTTP1.1 的请求消息和响应消息中支持 host头域，如果没有host头域会抛出一个错误 400。

> HTTP1.1 中 1.新增了某些字段，像强缓存和协商缓存的字段 `Expires`改成了`cache-control`，`last-Modified` 改成 `Etag`。2. 新增24个状态码, 410 请求的资源已经再服务器上被永久的删除了。3.新增 5 种请求的方式 put，delete，option，connect，trace。3. 增加了长连接`connection: keep-alive`。

HTTP.2 有哪些改进？/HTTP2.0 和 HTTP1.x 的区别

- 新的二进制格式，HTTP1.x 的解析基于文本协议，文本协议的格式存在多样性，比如音频，图片。但是二进制的就是0和1的组合，有很强的适用性。
- 头部压缩 header。在 `HTTP1.X` 中 header 有大量的重复信息，每次发送是都会携带这些重复信息。在 HTTP2.0 中使用 encode 来减少传输的 header 大小。
- 新增服务端推送。比如，浏览器只请求了index.html，但是服务器把index.html、style.css、example.png 全部发送给浏览器
- 多路复用。在 HTTP1.0 中每一次请求都需要TCP连接，请求结束就关闭。HTTP1.1 中增加了长连接，但是多个请求需要排队，等待前一个请求的结果返回后，后一个请求才可以发送。一旦某个请求超时就会造成阻塞。HTTP2.0 解决了这样的一个问题，多个请求可以在一个连接上并行执行，且互不干扰。

> 采用二进制的传输格式代替HTTP1.x中的文本传输协议，二进制传输格式具有很强的适用性 对头部压缩，在 HTTP1.x 中的 header 头部有很多的重复信息。HTTP2.0对头部进行了压缩处理 实现了多路复用。实现多个请求可以并行执行不需要等待 新增服务端推送功能。比如，浏览器只请求了index.html，但是服务器把index.html、style.css、example.png全部发送给浏览器

HTTP面试题

http面试答案HTTP协议 看完这篇HTTP，跟面试官扯皮就没问题了 面试 HTTP ，99% 的面试官都爱问这些问题