HTTP请求报文由四个部分组成：请求行、请求头部、空行、请求数据。

1.请求行

请求行由请求方法字段、URL字段和HTTP协议版本字段3个字段组成，它们用空格分隔。

2请求头

请求头部通知服务器有关于客户端请求的信息，典型的请求头有：

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。

它的作用是通过一个空行，告诉服务器请求头部到此为止。

3.请求数据

若方法字段是GET，则此项为空，没有数据

若方法字段是POST,则通常来说此处放置的就是要提交的数据

HTTP响应报文也由三部分组成：响应行、响应头、响应体

1.响应行

响应行一般由协议版本、状态码及其描述组成 比如 HTTP/1.1 200 OK

2.响应头

响应头用于描述服务器的基本信息，以及数据的描述，服务器通过这些数据的描述信息，可以通知客户端如何处理等一会儿它回送的数据。

3.响应体

响应体就是响应的消息体，如果是纯数据就是返回纯数据，如果请求的是HTML页面，那么返回的就是HTML代码，如果是JS就是JS代码，如此之类。

CSRF（Cross-site request forgery），中文名称：跨站请求伪造

以你名义发送邮件，发消息，盗取你的账号，甚至于购买商品

攻击者盗用了你的身份，以你的名义发送恶意请求。CSRF能够做的事情包括：以你名义发送邮件，发消息，盗取你的账号，甚至于购买商品，虚拟货币转账......造成的问题包括：个人隐私泄露以及财产安全。

两个条件：

1.登录受信任网站A，并在本地生成Cookie。

2.在不登出A的情况下，访问危险网站B。

CSRF攻击是源于WEB的隐式身份验证机制！WEB的身份验证机制虽然可以保证一个请求是来自于某个用户的浏览器，但却无法保证该请求是用户批准发送的！

防御：服务端的CSRF方式方法很多样，但总的思想都是一致的，就是在客户端页面增加伪随机数。为每个表单添加令牌token并验证。在请求中放入攻击者所不能伪造的信息，并且该信息不存在于Cookie之中。

鉴于此，我们将为每一个表单生成一个随机数秘钥，并在服务器端建立一个拦截器来验证这个token，如果请求中没有token或者token内容不正确，则认为可能是CSRF攻击而拒绝该请求。

（1）服务器发送给客户端一个token；

（2）客户端提交的表单中带着这个token。

（3）如果这个 token 不合法，那么服务器拒绝这个请求。



XSS攻击全称跨站脚本攻击

1. 通过document.cookie盗取cookie
2. 使用js或css破坏页面正常的结构与样式
3. 流量劫持（通过访问某段具有window.location.href定位到其他页面）

它指的是恶意攻击者往Web页面里插入恶意脚本代码，当用户浏览该页之时，嵌入其中Web里面的脚本代码会被执行，从而达到恶意攻击用户的特殊目的。

本质都是注入一段恶意代码，只是触发解析执行的方式不一样。

反射型XSS、存储型XSS、DOM型XSS。

**反射型 XSS**

反射型XSS，也叫非持久型XSS，是指发生请求时，XSS代码出现在请求URL中，作为参数提交到服务器，服务器解析并响应。响应结果中包含XSS代码，最后浏览器解析并执行。（没有对返回的数据进行解码和过滤等操作。）

**存储型 XSS**  
存储型XSS，也叫持久型XSS，主要是将XSS代码发送到服务器（不管是数据库、内存还是文件系统等。），然后在下次请求页面的时候就不用带上XSS代码了。

最典型的就是留言板XSS。用户提交了一条包含XSS代码的留言到数据库。当目标用户查询留言时，那些留言的内容会从服务器解析之后加载出来。浏览器发现有XSS代码，就当做正常的HTML和JS解析执行。XSS攻击就发生了。

**DOM XSS**  
DOM XSS攻击不同于反射型XSS和存储型XSS，DOM XSS代码不需要服务器端的解析响应的直接参与，而是通过浏览器端的DOM解析。这完全是客户端的事情。

可能触发DOM型XSS的属性：

document.referer属性

window.name属性

location属性

innerHTML属性

documen.write属性

**对cookie的保护**

1. 对重要的cookie设置httpOnly, 防止客户端通过document.cookie读取cookie。服务端可以设置此字段。

**对用户输入数据的处理**

1. 编码：不能对用户输入的内容都保持原样，对用户输入的数据进行字符实体编码。对于字符实体的概念可以参考文章底部给出的参考链接。
2. 解码：原样显示内容的时候必须解码，不然显示不到内容了。
3. 过滤：把输入的一些不合法的东西都过滤掉，从而保证安全性。如移除用户上传的DOM属性，如onerror，移除用户上传的Style节点，iframe, script节点等。

跨域：

同源策略限制了从同一个源加载的文档或脚本如何与来自另一个源的资源进行交互。这是一个用于隔离潜在恶意文件的重要安全机制。

所谓同源是指，域名，协议，端口均相同。

三种解决方案：jsonp,cors,反向代理nginx

Cookies、session、localStore、sessionStroe

HTTP是一种无状态的协议，不知道链接是谁发起的，所以需要浏览器把登录信息存起来，Session和Cookie就是为解决这个问题而提出来的两个机制。

1. Cookie（4K）是客户端上存储的小段文本，并随每一个请求发送至同一服务器，是在客户端保持状态的方案。有时也用Cookies，指某些网站为了辨别用户身份，进行session跟踪而存储在本地终端上的数据，通常经过加密。一般应用最典型的案列就是判断注册用户是否已经登过该网站。Cookie的主要内容包括：名字，值，过期时间，路径和域。

过期时间如不设，则浏览器关掉就消失，存储在内存当中；否则就按设置的时间来存储在硬盘上的，过期后自动清除。

路径和域就是对应的域名，a网站的cookie自然不能给b用。

1. Session存储在服务端，每建立一个连接就会生成一个Session ID，并将其通过响应发送到浏览器。浏览器把Session ID存在Cookie中，浏览器第二次发送请求会将前一次服务器响应中的Session ID放在请求中一并发送到服务器上，服务器从请求中提取出Session ID，并和保存的所有Session ID进行对比，找到对应的存放用户数据的类HashTable结构。

当用户在应用程序的 Web页间跳转时，存储在 Session 对象中的变量不会丢失而是在整个用户会话中一直存在下去。一般这个值会有个时间限制，超时后毁掉这个值，默认30分钟。

1、大小Cookie（4K）,Session（大）

2、Cookies客户端，Session服务端

3、Cookies不安全，容易被伪造，Session安全。

3.LocalStorage 和SessionStorage：两者都是仅在客户端（即浏览器）中保存，不参与和服务器的通信；

localStorage - 没有时间限制的数据存储

sessionStorage - 针对一个会话的数据存储,当用户关闭浏览器窗口后，数据会被删除。

Cookie 与SessionStorage 、LocalStorage 的区别(共同点：都是保存在浏览器端，且同源的。)

Cookie数据始终在同源的http请求中携带（即使不需要），即Cookie在浏览器和服务器间来回传递；Cookie数据还有路径（path）的概念，可以限制Cookie只属于某个路径下。存储大小限制也不同，Cookie数据不能超过4k，同时因为每次http请求都会携带Cookie，所以Cookie只适合保存很小的数据，如会话标识。而SessionStorage和LocalStorage不会自动把数据发给服务器，仅在本地保存。SessionStorage和LocalStorage 虽然也有存储大小的限制，但比Cookie大得多，可以达到5M或更大。

数据有效期不同，SessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie只在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭。

作用域不同，sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；localStorage 在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的。Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。Web Storage 的 api 接口使用更方便。

Put get

get是用来从服务器上获得数据，而Post是用来向服务器上传递数据。

jsonp,

cors,

反向代理nginx

https

状态码



301永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替

302 临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI

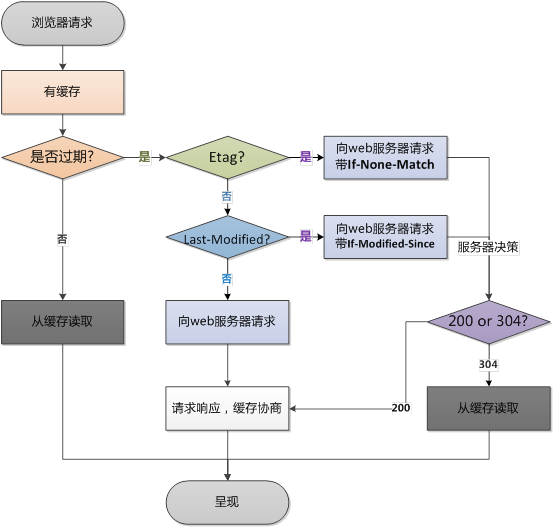
304 Not Modified 未修改。

所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。然后去缓存中读取资源。

缓存

403 Forbidden是HTTP协议中的一个状态码(Status Code)。可以简单的理解为没有权限访问此站。该状态表示服务器理解了本次请求但是拒绝执行该任务，该请求不该重发给服务器。

404服务器无法根据客户端的请求找到资源（网页）



1.强缓存

与强缓存相关的HTTP header 的字段有两个 Expires以及Cache-Control。

cache-control（优先级高）



其中最常用的值max-age单位为秒，对比expires体现着一个相对时间,即多少秒后这个强缓存机制下的缓存资源失效。

Expires

expires 字段规定了缓存的资源的过期时间，在此时间之前，缓存中的资源都是有效的，该字段的 value 是一个格林威治时间格式（GMT)的时间,即世界标准时间，js 通过 new Date().toUTCString()可得到，形如 Tue, 27 Feb 2018 06:37:48 GMT。他的缺点很明显，时间期限是服务器生成，存在着客户端和服务器的时间误差，固定时间，HTTP 1.0时的规范。相比较接下来介绍的cache-control优先级较低。

1. 协商缓存

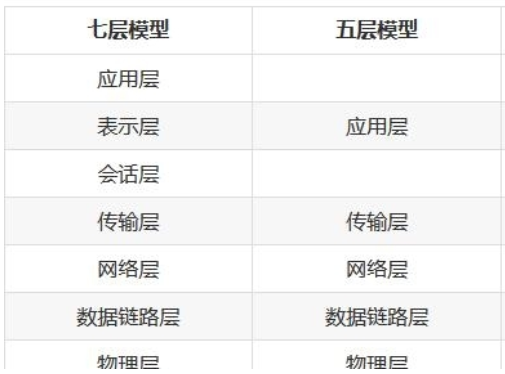
服务器端的唯一标识（优先级高）

Etag （响应头部）和 If-None-Match（请求头部）

最后修改的时间

Last-Modified（响应头部） 和 If-Modified-Since（请求头部）

http,tcp,udp简介及区别（网络传输协议）



HTTP：超文本传输协议(HyperText Transfer Protocol)是一种无状态协议，就是说客户端发送一次请求，服务器端接收请求，经过处理返回给客户端信息，然后客户端和服务器端的链接就断开了，为了维护他们之间的链接，让服务器知道这是前一个用户发送的请求，必须在一个地方保存客户端的信息，有2中解决方案，一是在客户端保存，二是在服务器端保存。

1.在客户端保存:Cookie

2.在服务器端保存:Session(session需要依靠cooke来实现)

3.在用户禁用cookie的限制下，只能使用URL重写的方式在每次请求之后附上一个键值对来保存客户端的信息。

4.隐藏表单。

HTTPS：（全称：Hypertext Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。

### HTTP与HTTPS的区别：

ttp和https使用的是完全不同的连接方式,用的端口也不一样,前者是80,后者是443。http的连接很简单,是无状态的,...

HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议 要比http协议安全。

tcp：（Transmission Control Protocol，传输控制协议）。tcp是面向连接的协议，也就是说，在收发数据前，必须和对方建立可靠的连接。一个TCP连接必须要经过三次“对话”才能建立起来。

### 层级：tcp属于传输层

### TCP建立过程（三次握手）

1 主机A通过向主机B 发送一个含有同步序列号的标志位的数据段给主机B ,向主机B 请求建立连接,通过这个数据段,主机A告诉主机B 两件事:我想要和你通信;你可以用哪个序列号作为起始数据段来回应我.

2 主机B 收到主机A的请求后,用一个带有确认应答(ACK)和同步序列号(SYN)标志位的数据段响应主机A,也告诉主机A两件事:我已经收到你的请求了,你可以传输数据了;你要用哪佧序列号作为起始数据段来回应我

3 主机A收到这个数据段后,再发送一个确认应答,确认已收到主机B 的数据段:"我已收到回复,我现在要开始传输实际数据了"。

这样3次握手就完成了,主机A和主机B 就可以传输数据了.

### tcp断开过程（四次握手）

TCP建立连接要进行3次握手,而断开连接要进行4次

1 当主机A完成数据传输后,将控制位FIN置1,提出停止TCP连接的请求

2 主机B收到FIN后对其作出响应,确认这一方向上的TCP连接将关闭,将ACK置1

3由B 端再提出反方向的关闭请求,将FIN置1

4 主机A对主机B的请求进行确认,将ACK置1,双方向的关闭结束.

由TCP的三次握手和四次断开可以看出,TCP使用面向连接的通信方式,大大提高了数据通信的可靠性,使发送数据端和接收端在数据正式传输前就有了交互,为数据正式传输打下了可靠的基础。

UDP：（User Data Protocol，用户数据报协议）

（1） UDP是一个非连接的协议，传输数据之前源端和终端不建立连接，当它想传送时就简单地去抓取来自应用程序的数据，并尽可能快地把它扔到网络上。在发送端，UDP传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽的限制；在接收端，UDP把每个消息段放在队列中，应用程序每次从队列中读一个消息段。

（2） 由于传输数据不建立连接，因此也就不需要维护连接状态，包括收发状态等，因此一台服务机可同时向多个客户机传输相同的消息。

（3） UDP信息包的标题很短，只有8个字节，相对于TCP的20个字节信息包的额外开销很小。

（4） 吞吐量不受拥挤控制算法的调节，只受应用软件生成数据的速率、传输带宽、源端和终端主机性能的限制。

（5）UDP使用尽最大努力交付，即不保证可靠交付，因此主机不需要维持复杂的链接状态表（这里面有许多参数）。

（6）UDP是面向报文的。发送方的UDP对应用程序交下来的报文，在添加首部后就向下交付给IP层。既不拆分，也不合并，而是保留这些报文的边界，因此，应用程序需要选择合适的报文大小。

我们经常使用“ping”命令来测试两台主机之间TCP/IP通信是否正常，其实“ping”命令的原理就是向对方主机发送UDP数据包，然后对方主机确认收到数据包，如果数据包是否到达的消息及时反馈回来，那么网络就是通的。

TCP：面向连接、传输可靠(保证数据正确性,保证数据顺序)、用于传输大量数据(流模式)、速度慢，建立连接需要开销较多(时间，系统资源)。

UDP：面向非连接、传输不可靠、用于传输少量数据(数据包模式)、速度快。

通俗一点说就是：TCP/IP管发管到，UDP管发不管到。因此，在安全性方面来说，TCP/IP更具有优越性。

输入URl地址过程：

输入网址

Url=网址

缓存解析

先去缓存当中看看有没有

域名解析

在发送http之前，需要进行DNS解析即域名解析。   
DNS解析:域名到IP地址的转换过程。域名的解析工作由DNS[服务器](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/qq_24147051/article/details/_blank)完成。解析后可以获取域名相应的IP地址

连接，三次握手

发送请求

浏览器向web服务器发送http请求；

服务器响应请求

服务器收到浏览器发送的请求信息，返回一个响应头和一个响应体。

页面渲染

浏览器收到服务器发送的响应头和响应体，进行客户端渲染，生成Dom树、解析css样式、js交互。