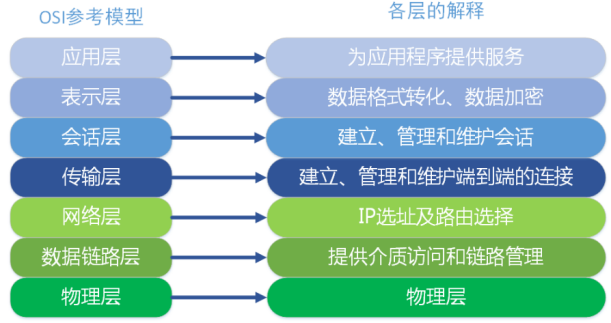
**TCP协议**

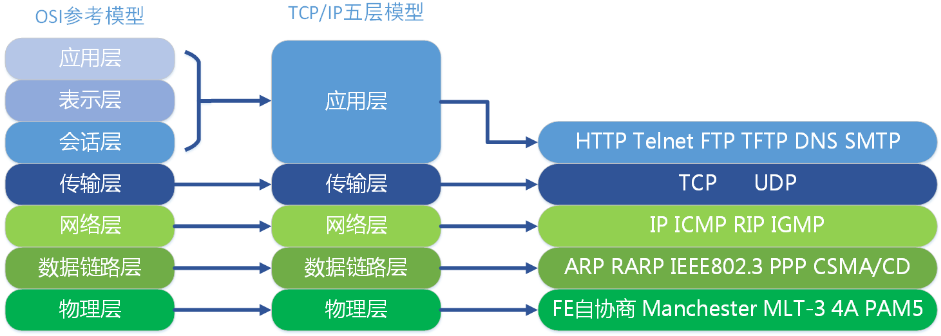
**OSI七层网络模型**

OSI定义了网络互连的七层框架（物理层、数据链路层、网络层、传输层、会话层、表示层、应用层），即ISO开放互连系统参考模型。



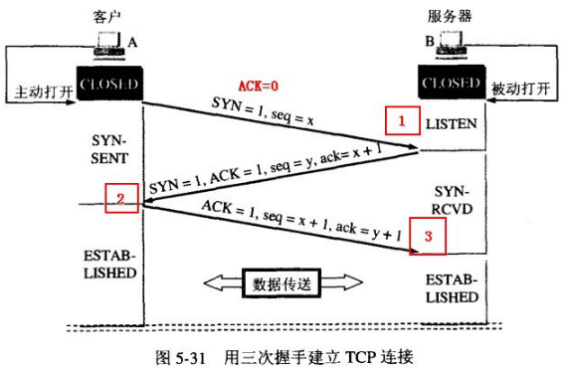
**TCP/IP五层模型**

说回TCP，简单说TCP（Transmission Control Protocol）即传输控制协议，是一种面向连接的、可靠的、基于ip的传输层协议。



**TCP三次握手**

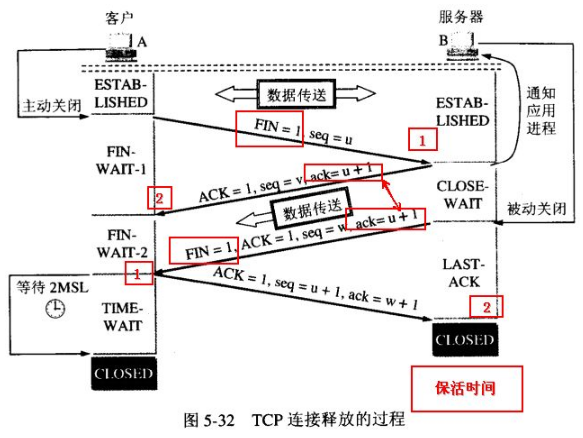
TCP三次握手其实就是TCP连接建立的过程，三次握手的目的是同步连接双方的序列号和确认号并交换TCP窗口大小信息



1. 客户端主动打开，发送连接请求报文段，将SYN标识位置为1，Sequence Number置为x（TCP规定SYN=1时不能携带数据，x为随机产生的一个值），然后进入SYN\_SEND状态
2. 服务器收到SYN报文段进行确认，将SYN标识位置为1，ACK置为1，Sequence Number置为y，Acknowledgment Number置为x+1，然后进入SYN\_RECV状态，这个状态被称为半连接状态
3. 客户端再进行一次确认，将ACK置为1（此时不用SYN），Sequence Number置为x+1，Acknowledgment Number置为y+1发向服务器，最后客户端与服务器都进入ESTABLISHED状态

**TCP四次挥手**

TCP四次挥手则是TCP连接释放的过程



1. 客户端发送一个报文给服务端（没有数据），其中FIN设置为1，Sequence Number置为u，客户端进入FIN\_WAIT\_1状态

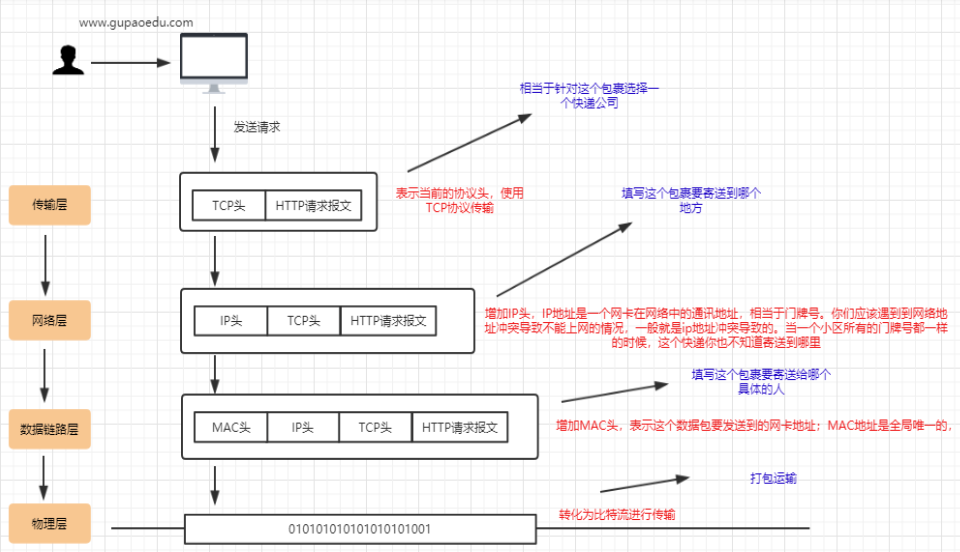
2. 服务端收到来自客户端的请求，发送一个ACK给客户端，Acknowledge置为u+1，同时发送Sequence Number为v，服务端年进入CLOSE\_WAIT状态

3. 服务端发送一个FIN给客户端，ACK置为1，Sequence置为w，Acknowledge置为u+1，用来关闭服务端到客户端的数据传送，服务端进入LAST\_ACK状态

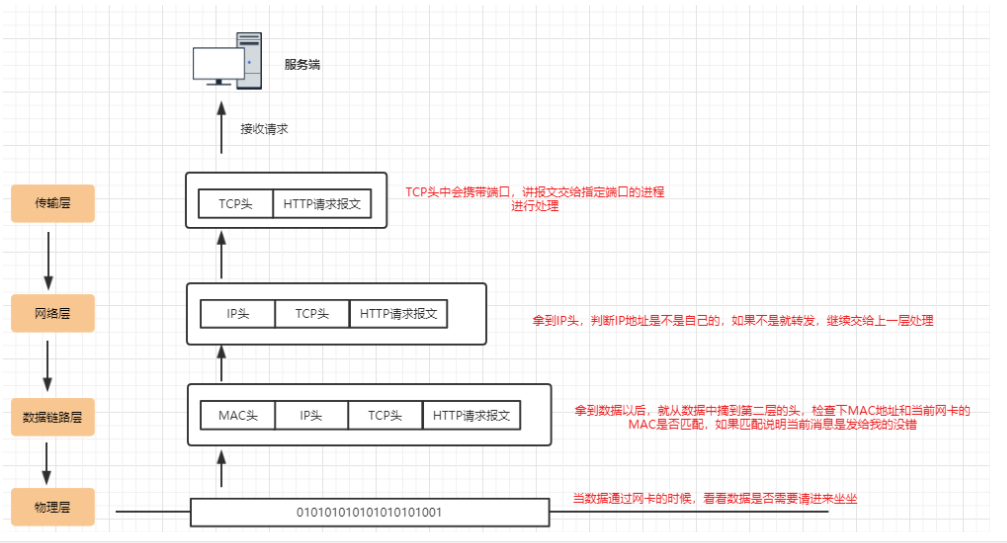
4. 客户端收到FIN后，进入TIME\_WAIT状态，接着发送一个ACK给服务端，Acknowledge置为w+1，Sequence Number置为u+1，最后客户端和服务端都进入CLOSED状态

**一个HTTP请求，在整个网络的请求过程**

当应用程序用TCP传送数据时，数据被送入协议栈中，然后逐个通过每一层直到被当作一串比特流送入网络。其中每一层对收到的数据都要增加一些首部信息



当目的主机收到一个以太网数据帧时，数据就开始从协议栈中由底向上升，同时去掉各层协议加上的报文首部。每层协议盒都要去检查报文首部中的协议标识，以确定接收数据的上层协议。这个过程称作分用



**什么是协议**

协议相当于两个需要通过网络通信的程序达成的一种约定，它规定了报文的交换方式和包含的意义。比如（HTTP）为了解决在服务器之间传递超文本对象的问题，这些超文本对象在服务器中创建和存储，并由Web浏览器进行可视化，完成用户对远程内容的感知和体验

**什么是IP协议**

T C P和U D P是两种最为著名的传输层协议，他们都是使用I P作为网络层协议。IP协议提供了一组数据报文服务，每组分组报文都是由网络独立处理和分发，就像寄送快递包裹一样，为了实现这个功能，每个IP报文必须包含一个目的地址的字段；就像我们寄送快递都需要写明收件人信息，但是和我们寄送快递一样，也可能会出现包裹丢失问题，所以IP协议只是一个“尽力而为”的协议，在网络传输过程中，可能会发生报文丢失、报文顺序打乱，重复发送的情况。IP协议层之上的传输层，提供了两种可以选择的协议，TCP、UPD。这两种协议都是建立在IP层所提供的服务基础上，根据应用程序的不同需求选择不同方式的传输

**TCP/IP**

TCP协议能够检测和恢复IP层提供的主机到主机的通信中可能发生的报文丢失、重复及其他错误。TCP提供了一个可信赖的字节流通道，这样应用程序就不需要考虑这些问题。同时，TCP协议是一种面向连接的协议，在使用TCP进行通信之前，两个应用程序之间需要建立一个TCP连接，而这个连接又涉及到两台电脑需要完成握手消息的交换。

**UDP/IP**

UDP协议不会对IP层产生的错误进行修复，而是简单的扩展了IP协议“尽力而为”的数据报文服务，使他能够在应用程序之间工作，而不是在主机之间工作，因此使用UDP协议必须要考虑到报文丢失，顺序混乱的问题

**HTTP**

**HTTP是什么**

HTTP，英文全称为HyperText Transfer Protocol，即超文本传输协议，是互联网上应用最为广泛的一种网络协议。HTTP是一种属于应用层的面向对象的协议，一次HTTP操作称为一个实物

工作过程：

1、首先客户端与服务器需要建立连接，只需要单击某个超级链接，HTTP的工作开始

2、建立连接后，客户端发送一个请求给服务器（默认请求服务器的80端口），请求方式的格式为:统一资源标识符（URL）+协议版本号+MIME信息包括请求修饰符、客户端信息和可能的内容

3、服务器接收到请求后，给予响应的相应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后面是MIME信息包括服务器信息、实体信息和可能的内容

4、客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户端和服务器断开连接

特点：

1、支持客户端/服务器模式

2、简单快速，客户端向服务器发送请求服务时，只需要传送请求方法和路径。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度快

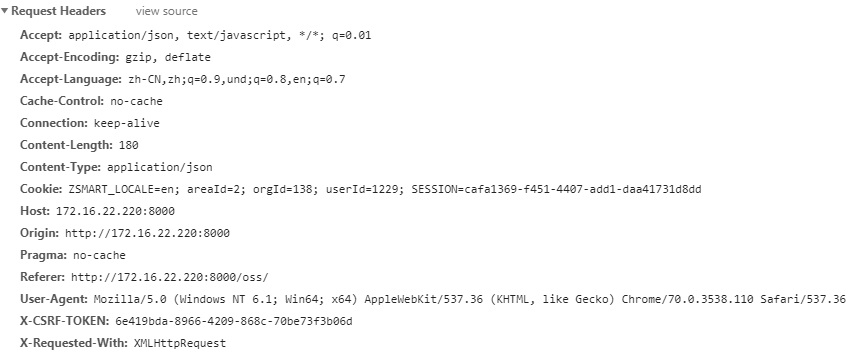
3、灵活，HTTP允许传输任意类型的数据对象，正在传输的类型由Content-Type加以标记

4、无连接，无连接的含义是限制每次连接只处理一个请求，服务器处理完成客户端的请求，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

5、无状态，HTTP协议是无状态协议，无状态是指协议对于事物处理没有记忆能力，血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输的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快

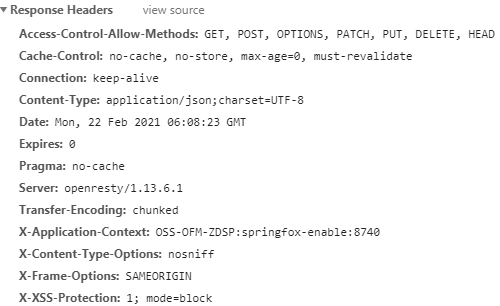
**HTTP请求头**





**HTTP响应头**





**HTTP状态码**



**HTTP缓存**

缓存分为服务端缓存和客户端缓存，客户端缓存一般指的是浏览器缓存，目的就是加速各种静态资源的访问。浏览器缓存有两种机制：HTML Meta标签和HTTP HEADER信息。

1. HTML Meta

HTML页面的<head>节点中加入<meta>标签，代码如下：



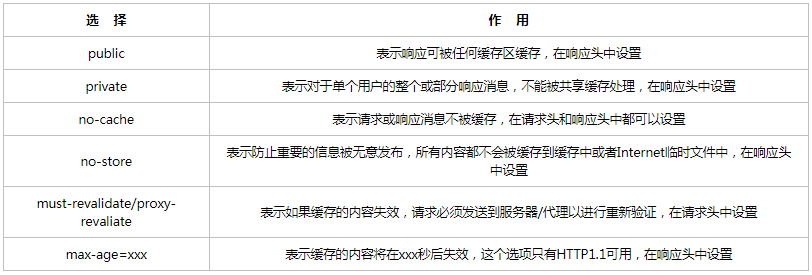
上述代码的作用是告诉浏览器当前页面不被缓存，每次访问都需要去服务器读取

1. HTTP HEADER
2. 、Expires策略

Expires策略是Web服务器响应消息头字段，在响应HTTP请求时告诉浏览器在过期时间前浏览器可以直接从浏览器缓存取数据而无须再次请求。不过Expires策略在HTTP1.1基本忽略，因为Expires返回的到期时间是服务器的时间，如果客户端和服务器的时间相差很大，那么误差就很大。

1. 、Cache-Control策略

Cache-Control策略与Expires策略的作用一致，都是致命当前资源的有效期，控制浏览器是否直接从浏览器缓存取数据还是重新发送请求到服务器取数据。只不过Cache-Control的选择更多、被各浏览器支持得更好、设置得更细致，如果同时设置的话，其优先级也比较高（高于Expires）



**HTTPS**

**HTTPS的概念**

HTTPS全称为Hypertext Transfer Protocol over Secure Socket Layer，及以安全为目标的HTTP通道，简单说就是HTTP的安全版本。

HTTPS其实是由两部分组成的：HTTP+TLS/SSL，即HTTP下加入TLS/SSL层，HTTPS的安全基础就是TLS/SSL。服务端和客户端的信息传输都会通过TLS/SSL进行加密，所以传输的数据都是加密之后的数据。TLS的前身就是SSL协议，因此没有特别说明TLS/SSL说的都是同一个东西。

**HTTP在安全方面的缺陷**

HTTP本身是明文传输的，没有经过任何安全处理。

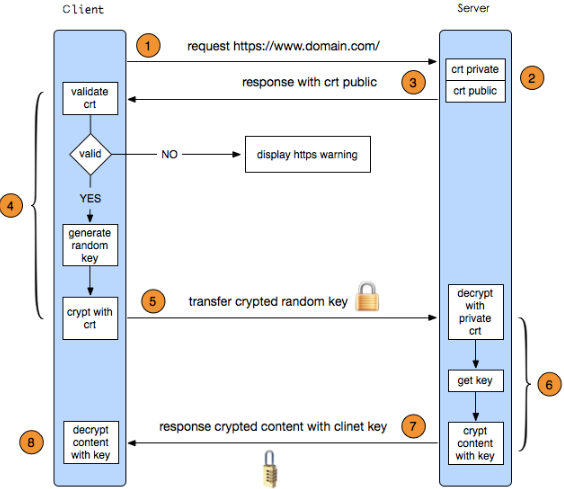
HTTPS协议提供了三个强大的功能来对抗上述的劫持行为：

1、内容加密。浏览器到百度服务器的内容都是以加密形式传输的，中间者无法直接查看原始内容

2、身份认证。保证用户访问的是百度服务，即使被DNS劫持到了第三方站点，也会提醒用户没有访问百度服务

3、数据完整。防止内容被第三方冒充或篡改

**HTTPS工作原理**



工作过程：

1. 客户端发起HTTPS请求

浏览器里面输入一个HTTPS网址，然后连接到服务端的443端口上。注意这个过程中客户端会发送一个密文族给服务端，密文族是浏览器所支持的加密算法的清单。

1. 服务端配置

采用HTTPS协议的服务器必须要有一套数字证书，可以自己制作，也可以向组织申请。区别就是自己颁发的证书需要客户端验证通过才可以继续访问，而使用受信任的公司申请的证书则不会弹出提示页面。

这套证书其实就是一对公钥和私钥，可以这么理解，公钥就是一把锁头，私钥就是这把锁的钥匙，锁头可以给别人对某个东西进行加锁，但是加锁完毕之后，只有持有这把锁的钥匙才可以解锁看到加锁的内容。

1. 传送证书

这个证书其实就是公钥，只是包含了很多信息，如证书的颁发机构、过期时间等等。

1. 客户端解析证书

这部分工作是由客户端的TLS来完成的，首先会验证公钥是否有效，如颁发机构、过期时间等等，如果发现异常则会弹出一个警告框，提示证书存在问题。如果证书没有问题，那么就生成一个随机值，然后用证书对该随机值进行加密。

注意一下上面提到的"发现异常"。证书中会包含数字签名，该数字签名是加密过的，是用颁发机构的私钥对本证书的公钥、名称及其他信息做hash散列加密而生成的。客户端浏览器会首先找到该证书的根证书颁发机构，如果有，则用该根证书的公钥解密服务器下发的证书，如果不能正常解密，则就是"发现异常"，说明该证书是伪造的。

1. 传送加密信息

这部分传送的是用证书加密后的随机值，目的就是让服务端得到这个随机值，然后客户端和服务端的通信就可以通过这个随机值来进行加密和解密了

1. 服务端解密信息

服务端用私钥解密后，得到了客户端传过来的随机值，至此一个非对称加密的过程结束，看到TLS利用非对称加密实现了身份认证和密钥协商。然后把内容通过该值进行对称加密。

1. 传输加密后的信息

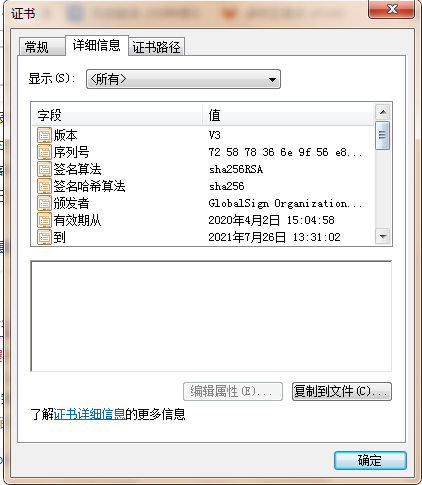
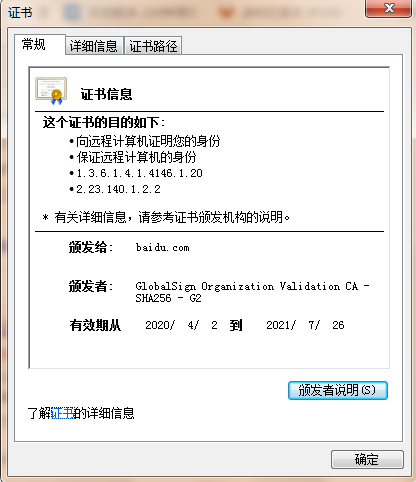
这部分是服务端用随机值加密后的信息，可以在客户端被还原。

1. 客户端解密信息

客户端用之前生成的随机值解密服务端传送过来的信息，于是获取了解密后的内容，至此一个对称加密的过程结束，看到对称加密是用于对服务器待传送给客户端的数据进行加密用的。整个过程即使第三方监听了数据，也束手无策。

**证书**

证书其实就是服务器自己的一套身份信息，客户端拿到证书之后会首先验证证书的合法性，如颁发证书的机构是否合法、证书中包含的网站地址是否与正在访问的地址一致等，如果证书受信任，则浏览器栏里面会显示一个小锁头，否则则会给出证书不受信的提示。



**GET和POST**

HTTP定义了与服务器交互的不同方法，最基本的方法有4种，分别是GET、POST、PUT、DELETE。URL全称是资源描述符，我们可以这么认为：一个URL地址，它用于描述网络上的一个资源，而HTTP的GET、POST、PUT、DELETE就对应着对这个资源的查、改、增、删4个操作。

**原理上的区别**

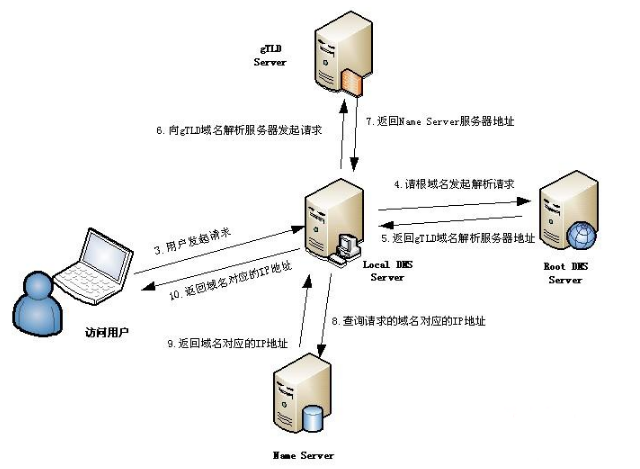
1. GET一般用于获取/查询资源信息，POST一般用于更新资源信息。
2. 、根据HTTP规范，GET用于信息获取，而且应该是安全和幂等的
3. 、所谓安全意味着该操作用于获取信息而不是修改信息，换句话说，GET请求一般不应该产生副作用，也就是说，它仅仅是获取资源信息，就像数据库查询一样，不会修改、增加数据，不影响资源的状态
4. 、所谓幂等意味着对同一个URL的多个请求应该返回相同的结果
5. 、根据HTTP规范，POST表示可能修改服务器上的资源的请求，比如一个新闻网站，读者对新闻资源发表的评论应该通过POST实现
6. 只有GET方式才能从浏览器缓存中存、取资源，POST方式则不可以。

**表象区别**

1. URL请求的方式为GET方式
2. 表单默认提交方式为GET方式
3. 表单的action中?（a=1）后面带了参数，只有POST方式可以取到（request.getParameter("a");）
4. 表单提交，GET方式会把所有控件值都带在action后面，即使是password，也是明文形式的，POST方式则不会
5. POST提交表单比GET方式提交表单更安全。因为GET方式提交的数据，表单参数是在URL中的，而POST方式提交的数据，表单参数是在请求体中的

**DNS域名解析过程**

我们知道互联网都是通过URL来发布和请求资源的，而URL中的域名需要解析成IP地址才能与远程主机建立连接，如何将域名解析成IP地址就属于DNS解析的工作范畴。



1. 浏览器会检查缓存中有没有这个域名对应的解析过的IP地址，如果缓存中有，这个解析过程就结束。
2. 如果用户浏览器缓存中没有数据，浏览器会查找操作系统缓存中是否有这个域名对应的DNS解析结果。其实操作系统也有一个域名解析的过程，在Windows中可以通过C:\Windows\System32\drivers\etc\hosts文件来设置，在Linux中可以通过/etc/hosts文件来设置，用户可以将任何域名解析到任何能够访问的IP地址。
3. 前两个过程无法解析时，就要用到我们网络配置中的"DNS服务器地址"了。操作系统会把这个域名发送给这个LDNS，也就是本地区的域名服务器。这个DNS通常都提供给用户本地互联网接入的一个DNS解析服务，例如用户是在学校接入互联网，那么用户的DNS服务器肯定在学校；
4. 如果LDNS仍然没有命中，就直接到Root Server域名服务器请求解析
5. 根域名服务器返回给本地域名服务器一个所查询的主域名服务器（gTLD Server）地址。gTLD是国际顶级域名服务器，如.com、.cn、.org等，全球只有13台左右
6. 本地域名服务器LDNS再向上一步返回的gTLD服务器发送请求
7. 接受请求的gTLD服务器查找并返回此域名对应的Name Server域名服务器的地址，这个Name Server通常就是用户注册的域名服务器，例如用户在某个域名服务提供商申请的域名，那么这个域名解析任务就由这个域名提供商的服务器来完成
8. Name Server域名服务器会查询存储的域名和IP的映射关系表，在正常情况下都根据域名得到目标IP地址，连同一个TTL值返回给DNS Server域名服务器
9. 返回该域名对应的IP和TTL值，LDNS会缓存这个域名和IP的对应关系，缓存时间由TTL值控制
10. 把解析的结果返回给用户，用户根据TTL值缓存在本地系统缓存中，域名解析过程结束

**清除缓存的域名**

我们知道DNS域名解析后会缓存解析结果，其中主要在两个地方缓存：

1、Local DNS Server

2、用户的本地机器

这两个缓存都是TTL值和本机缓存大小控制的，但是最大缓存时间是TTL值，基本上Local DNS Server的缓存时间就是TTL控制的，很难人工介入

1. Windows环境下可以再命令行执行ipconfig /flushdns命令来刷新缓存
2. Linux环境下可以通过/etc/init.d/nscd restart来清除缓存
3. Java应用中JVM也会缓存DNS的解析结果，这个缓存是在InetAddress类中完成的

它有两种缓存策略：

1）、正确解析结果缓存

2）、失败解析结果缓存

这两个缓存时间有两个配置项控制，配置项在%JAVA\_HOME%\lib\security\java.security文件中配置的，这两个配置项分别是networkaddress.cache.ttl和networkaddress.cache.negative.ttl，它们的默认值分别是-1（永不失效）和10（缓存10秒），直接修改这两个值就可以了，也可以通过在Java启动参数中增加-Dsun.net.inetaddr.ttl=xxx来修改默认值，也可以通过InetAddress类动态修改。

**几种域名解析方式**

域名解析记录主要分为A记录、MX记录、CNAME记录、NS记录和TXT记录：

1. A记录

A代表Address，用来指定域名对应的IP地址，如将item.taobao.com指定到115.238.23.xxx，将switch.taobao.com指定到121.14.24.xxx。A记录可以将多个域名解析到一个IP地址，但是不能将一个域名解析到多个IP地址

1. MX记录

Mail Exchange，就是可以将某个域名下的邮件服务器指向自己的Mail Server，如taobao.com域名的A记录IP地址是115.238.25.xxx，如果将MX记录设置为115.238.25.xxx，即xxx@taobao.com的邮件路由，DNS会将邮件发送到115.238.25.xxx所在的服务器，而正常通过Web请求的话仍然解析到A记录的IP地址

1. CNAME记录

Canonical Name，即别名解析。所谓别名解析就是可以为一个域名设置一个或者多个别名，如将aaa.com解析到bbb.net、将ccc.com也解析到bbb.net，其中bbb.net分别是aaa.com和ccc.com的别名

1. NS记录

为某个域名指定DNS解析服务器，也就是这个域名由指定的IP地址的DNS服务器取解析

1. TXT记录

为某个主机名或域名设置说明，如可以为ddd.net设置TXT记录为"这是XXX的博客"这样的说明

**session和cookie**

**为什么要使用session和cookie**

为什么要使用session和cookie这个话题就要从HTTP状态协议的无状态性开始说起了。

无状态协议是指协议对事物处理没有记忆能力。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应答就很快。

**Cookie**

cookie是通过客户端保持状态的解决方案。从定义上说，cookie就是服务器发送给客户端的特殊信息，而这些信息以文本文件的方式存放在客户端，然后客户端每次向服务器发送请求的时候都会带上这些特殊的信息。

当用户使用浏览器访问一个支持cookie的网站的时候，会有如下步骤：

1、用户会提供包括用户名在内的个人信息并且提交至服务器

2、服务器在向客户端会传相应的超文本的同时，发回这些个人信息。当然这些信息并不是存放在HTTP响应体（Response Body）中的，而是存放在HTTP响应头（Response Header）中的

3、当客户端浏览器收到来自服务器的响应之后，浏览器会将这些信息存放在一个统一的位置

4、之后，客户端再向服务器发送请求的时候，都会把相应的cookie再次返回至服务器，而这次，cookie信息则存放在HTTP请求头中了

缺点：

1. 使用cookie来传递信息，随着cookie个数的增多和访问量的增加，它占用的网络带宽也很大
2. cookie并不安全，因为cookie是存放在客户端的，所以这些cookie可以被访问到，设置可以通过插件添加、修改cookie。

**Session**

session是相对于cookie的另外一个解决方案，它是通过服务器来保持状态的。session指的是服务器为客户端所开辟的存储空间，在其中保存的信息就是用于保存状态的。

首先，session是服务器端程序运行的过程中创建的，不同语言实现的应用程序有不同创建session的方法。在创建了session的同时，服务器会为该session生成唯一的sessionId，而这个sessionId被创建了之后，就可以调用session相关的方法往session中增加内容了，而这些内容只会保存在服务器中，发送到客户端的只有sessionId。当客户端再次发送请求时，会将这个sessionId带上，服务器收到请求之后就会根据sessionId找到对应的session，从而再次使用。这样的一个过程，让用户的状态得以保持

其次，每个session都有一个sessionId，这个ID存放在哪里？有两种方式：

1、通过URL存取，URL会带上一个;jsessionId=xxxxxx等，这样每次重新请求的时候都传了sessionId给服务器

2、通过cookie存取（Tomcat默认如此），这种cookie是session cookie，区别于persistent cookies也就是我们常说的cookie，session cookie要注意的是存储在浏览器内存中（至于浏览器内存在哪里，这是和浏览器相关的，比方说我用的是360浏览器，那就在360se6\User Data\Default这个路径下）而不是写到硬盘上。程序一开始执行，服务器就生成一个sessionId并通过cookie携带客户端浏览器的缓存中，当下一次访问的时候，服务器先检测一下是否有这个cookie，如果有就取它的ID，如果没有就再生成一个。这就是为什么关闭浏览器之后，再进去session已经没有了，其实在服务器端session并没有清空，而是sessionId变了。

再者，浏览器关闭，session没有了的说法是不正确的。如果浏览器关闭，服务器保存的session数据不是立即释放的，此时数据还会存在，只要我们知道那个sessionId，就可以继续通过请求获得此session的信息。session里面的数据都放在服务器端，通过sessionId保证不会访问错误，服务端自动对session进行管理，如果在规定的时间内没有访问，则释放掉这个session。

最后提两点：

1、sessionId通常是看不到的，但是当我们把浏览器的cookie禁止之后，Web服务器会采用URL重写的方式传递sessionId，这样就可以在地址栏看到sessionId了

2、session cookie不可以跨窗口使用

缺点：

1. 不容易在多态服务器之间共享，这是致命的弱点
2. session存放在服务器中，所以session如果太多会非常消耗服务器的性能

**重定向和转发**

**区别**

1、从上面的一次重定向和一次转发的过程来看，我们可以得出第一个差别点，就是重定向是两次请求，转发是一次请求，因此转发的速度要快于重定向

2、第二个差别点也很明显，重定向之后地址栏上的地址会发生变化，变化成第二次请求的地址，转发之后地址栏上的地址不会变化，还是第一次请求的地址

**参数解读**

写法：

1. response.sendRedirect("XXX");

1）、重定向的时候，如果参数为"http://xxxxxxxxxx"，那么重定向到的是指定的重定向地址，无论该重定向地址是内部的还是外部的，但是这里可能出现跨域问题：

什么是跨域：

跨域（CORS）是指不同域名之间相互访问。跨域，指的是浏览器不能执行其他网站的脚本，它是由浏览器的同源策略所造成的，是浏览器对于JavaScript所定义的安全限制策略。

什么情况会跨域：

同一协议， 如http或https

同一IP地址, 如127.0.0.1

同一端口, 如8080

以上三个条件中有一个条件不同就会产生跨域问题

后端解决方案：

nginx反向代理解决跨域

服务端设置Response Header(响应头部)的Access-Control-Allow-Origin

在需要跨域访问的类和方法中设置允许跨域访问（如Spring中使用@CrossOrigin注解）；

继承使用Spring Web的CorsFilter（适用于Spring MVC、Spring Boot）

实现WebMvcConfigurer接口（适用于Spring Boot）

2）、如果参数不是"http://xxxxxxxxxx"的样子，而是一般的写法，那么都是Tomcat内部跳转（注意是Tomcat内部跳转而不是项目内部跳转）

（1）重定向，如果XXX以"/"开头，则表示目标地址为http://ip:port/XXX；如果不以"/"开头，则表示目标地值为http://ip:port/项目名/XXX

（2）请求转发，无论XXX是否以"/"开头，都跳转到http://ip:port/项目名/XXX

2、request.getRequestDispatcher("XXX").forward(request, response);

**重定向和请求转发如何选择**

1. 选择重定向作为页面跳转
2. 选择转发作为内部Servlet跳转