网络协议

网络模型

1. 网络是一个很复杂的系统，包含大量的应用程序、端系统、数据链路、分组交换机、各种协议；
2. 网络协议的层次：为了给网络协议的设计提供一个结构，设计者提出分层（layer）的方式组织协议，每一个协议属于多个层次模型中的其中一个，每一层都是向它的上一层提供服务，即服务模型。每一个分层中的所有协议成为协议栈。而因特网的协议栈由五个部分组成，由上到下分别是：应用层（应用程序）、传输层（为应用层提供通信服务）、网络层（为传输层提供数据传送服务）、链路层（为网络层提供数据在传输媒介中的规范）、物理层（为链路层提供物质媒介）
3. 因特网的协议栈分层内容：
   1. 应用层是应用程序和网络协议存放的分层，主要包括HTTP（超文本传输协议）、DNS（Domain Name System）域名系统，即域名解析协议、SMTP(电子邮件传输协议)、FTP（文件上传协议）。应用层协议分布在多个端系统上，一个端系统应用程序与另一个端系统应用程序交换信息分组，位于应用层的信息称为报文（message）
   2. 传输层在应用程序断点之间传送应用程序报文，主要有TCP（Transform Control Protocol）传输控制协议和UDP（User Datagram Protocol）用户数据报协议，利用这两者中的任何一个都可以传输报文，又称为报文段（segment）
   3. 网络层将称为数据报（datagram）的网络分层，从一台主机移动到另一台主机。有一个非常重要的协议即IP（Internet Protocol），所有具有网络层的因特网组件都必须运行这个协议，IP是一种网际协议，除了IP协议网络层还包含其他网际协议和路由选择协议，一般网络层统称IP协议
   4. 链路层，为了将网络分组从一个节点运输到另一个节点，必须依靠链路层提供服务，链路层主要是以太网、wifi、电缆接入的DOCSIS（有线电缆数据服务接口规范）协议，由于数据从原目的地传送通常要经过数个链路，一个数据包可能会沿途经过数个不同链路层协议的处理，所以链路层的分组为帧(frame)
   5. 物理层的作用是将帧中的一个个比特从一个节点运输到另一个节点，物理层仍然使用链路层协议，与实际的物理传输介质有关，如以太网有很多物理层协议：关于双绞铜线、同轴电缆、光纤等
   6. 图示：

传输层

网络层

链路层

应用层

物理层

1. OSI模型
   1. 相比计算机网络协议模型，OSI模型在应用层和传输层之间多出了两层，表示层和会话层。
      1. 表示层：数据压缩、数据加密、数据描述，数据描述使得应用程序不再担心计算机内部存储格式的问题
      2. 会话层：提供了数据交换的定界和同步功能，包括建立检查点和恢复方案
   2. 图示：

应用层

会话层

表示层

网络层

传输层

链路层

物理层

1. 名词解释
2. HTTP：即超文本传输协议（Hypertext Transfer Protocol）,进行文字拆分即：超文本、传输、协议
   1. 超文本：早期是指文本即一种能被解析的有意义的二进制数据包，现在包含了图片、视频、音频、超链接的跳转,即语义扩大后的文本被称为超文本
   2. 传输：两个端进行通信时，超文本作为二进制数据包，由传输载体（同轴电缆、电话线、光纤）将数据包由计算机终端传输到另一个终端的过程就称为传输。其中传输数据的是请求方，接受数据的是应答方，且二者之间可以相互转换
   3. 协议：即网络中传递、管理信息的一些规范，计算机之间相互通信的一些规定
   4. 优点：
      1. 简单灵活易扩展
      2. 应用广泛环境成熟
      3. 无状态：即服务器无记忆能力，不需要额外的资源来记录状态信息，减轻服务器压力，节省内存
   5. 缺点：
      1. 无状态：导致其无法完成需要多个步骤的事务操作，每一次都要询问身份信息，增加信息传输量
      2. 明文传输：协议里的报文即header部分不是以二进制方式传输的，所以易被监听和窥探，无法判断是否被修改过
      3. 性能
   6. 总结：HTTP是一个在计算机中专门在两点之间传输文字、图片、视频、音频等超文本数据约定和规范
3. Web浏览器：HTTP请求的发起方，用户在浏览器中输入网址，由DNS进行域名解析，完成网址到IP的映射，然后将请求发送到具体的服务器，再由服务器返回我们想要的内容，浏览器执行HTML编码，将结果显示在浏览器正文中
4. Web服务器(Web Sever)：一般是指网站服务器，是HTTP请求的应答方，目前最主流的web服务器是Apache、Nginx、IIS
5. CND:即内容分发网络（Content Delivery Network）,它应用了HTTP协议里的缓存和代理技术，代替源站响应客户端的请求。它是建立在现有网络基础之上的网络，依靠在各地的边缘服务器，通过中心平台的负载均衡、 内容分发、调度等功能，使用户就近获取资源，降低网络拥塞，提高响应速度和效率，关键技术是内容存储和分发技术
6. WAF:是一种Web应用程序防护系统（Web Application Firewall），通过执行一系列针对HTTP/HTTPS的安全策略为Web应用提供保护，是 应用层面的防火墙，专门检测HTTP流量，访华Web应用的安全技术。它 通常位于Web服务器之前，可以阻止SQL注入、跨站脚本等攻击
7. WebService:是一种Web应用程序，也是一种跨编程语言和跨操作平台的远程调用技术，由W3C制定的应用服务开发规范，使用client-server主从架构，通常使用WSDL定义服务接口，使用HTTP协议传输XML或 SOAP信息，是一个基于Web（HTTP）的服务架构技术，可以在内网和外网运行
8. HTML
9. Web页面构成
10. 与HTTP有关的协议
11. TCP/IP:一般称之为协议簇，因为其中不仅仅只有TCP协议和IP协议，它是一系列通信协议的统称。核心是TCP和IP协议，还包含UDP、ICMP、ARP，共同构成复杂而有层次的协议栈。
12. TCP:即传输控制协议（Transmission Control Protocol）,HTTP使用TCP作为通信协议，因为TCP可靠，保证数据不丢失
13. IP:即网络协议（Internet Protocol）,主要解决通信双发寻址的问题，IP协议使用ip地址来标识互联网上的一台计算机
14. DNS：即域名系统（Domain Name System）,把IP地址转换为便于人们记忆的协议，它作为将域名和IP地址相互映射的分布式数据库，以便人们访问互联网
15. URI/URL：
16. URI:即统一资源标识符（Uniform Resource Identifier）,使用它就可以唯一的标识互联网上的资源
17. URL:即统一资源定位符（Uniform Resource Locator）,俗称网址是URI的一个子集
18. URI中不仅包含着URL，还包括URN（统一资源名称），URL和URN同级各占一半
19. HTTPS：由于HTTP是明文传输很容易受到攻击，因此HTTPS应运而生。
20. 和HTTP的区别：HTTPS是以安全为目标的HTTP通道，在HTTP基础之上通过传输加密和身份认证保证了传输过程的安全性，在HTTP基础上加入了SSL层（安全套接层）。
21. HTTP报文详解
22. HTTP报文组成部分：
23. 起始行：描述请求或响应的基本信息
24. 头部字段：使用key-value更详细的说明报文
25. 消息正文：实际传输的数据，包含文本、视频、音频、图片等类型数据
26. HTTP请求标头根据上下文分为四种：通用标头、请求标头、响应标头、实体标头，根据代理分为两种：端到端标头、逐跳标头
27. 通用标头：Date、CacheControl、Connection
28. Connection：决定当前事务（一次三次握手和四次挥手）完成后，是否关闭此次连接，有两种值：keep-alive持久性连接（完成一次事务后连接不关闭），close非持久性连接（一次事务完成后关闭连接）
29. Keep-Alive:表示Connection非持续连接的存活时间，如：

Connection:keep-alive

Keep-Alive:timeout=5,max=997

timeout：表示空闲连接保持打开状态的最短时间（单位秒）

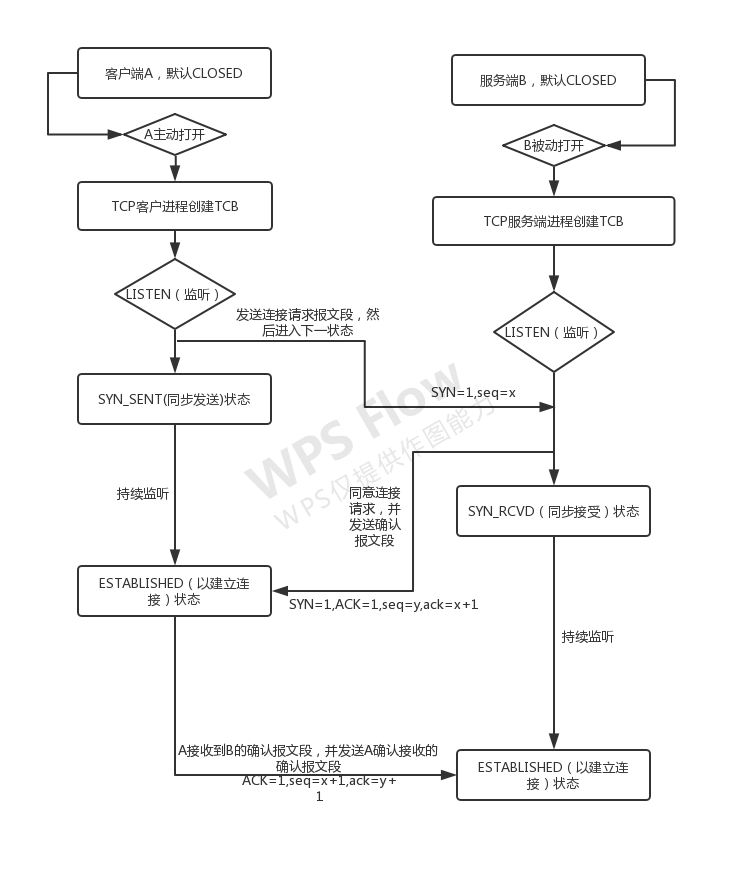
max：表示此连接在关闭之前允许的最大连接请求数

1. 请求标头：If-modified-since、If-none-match、Referer（发起请求的客户端页面地址）、Upgrade-Insecure-Requests、Host(服务器域名)
2. Transfer-Encoding：传输报文主体是所采用的编码方式，如：Transfer-Encoding:chunked。且http/1.1的传输编码方式仅对分块传输编码有效
3. 响应标头：Last-modified（资源最后一次修改的日期时间）、Etag、Expires(实体主体过期的日期时间)
4. X-Frame-Options：用于控制网站内容在其他web网站的frame标签中的显示问题，主要用于防止点击劫持攻击
5. Access-Control-Allow-Origin:指定一个来源，告诉浏览器允许此来源进行资源访问，也可使用通配符（\*）来进行处理，如果服务器指定单个来源时 ，还应在响应头字段Vary中包含Origin,以告诉客户端，服务器会根据原始请求标头的值而有所不同，如：

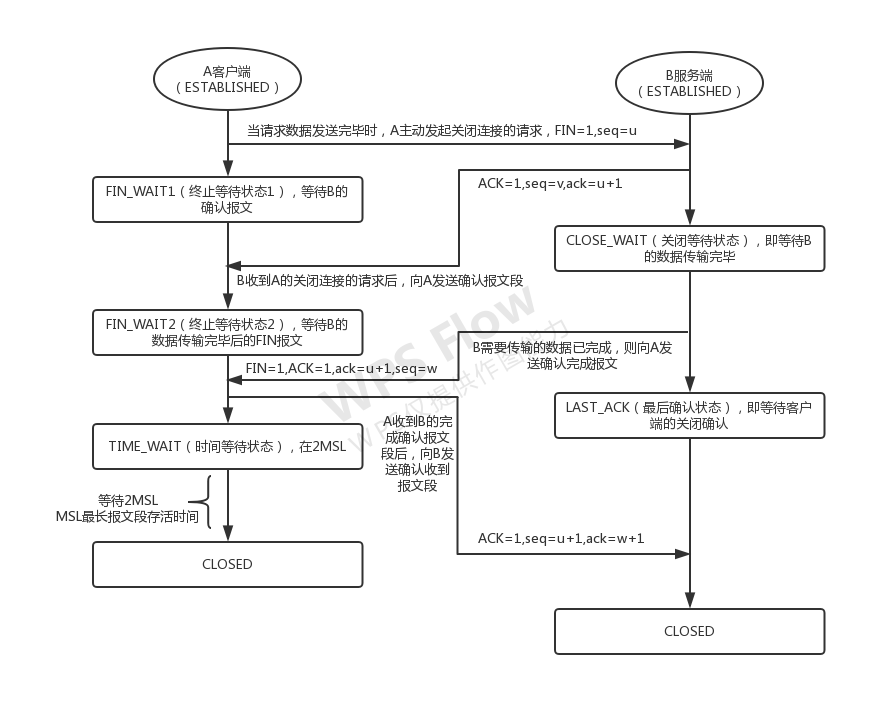
Access-Control-Allow-Origin: http://[www.baidu.com](http://www.baidu.com)

Vary:Origin

1. 实体标头：针对消息正文内容的标头，Content-length、Content-language、Content-Encoding（压缩媒体类型，指示对实体应用了何种编码，如：gzip\compress\deflate\identity）
2. 端到端标头（End-to-end）：这些标头必须发送给消息的最终接收者：请求的服务器或者响应的客户端，中间代理必须重新传输未经修改的标头，并且缓存必须存储这些信息
3. 逐跳标头(Hop-by-hop)：此类标头只对单次转发有效，会因通过缓存或代理而不再转发；http/1.1中有8种，其余都是端到端标头：Connection、Keep-Alive、Trailer、TE（流量工程，一套工具和方法）、Upgrade、Proxy-Authenticate、Proxy-Authorization、Transfer-Encoding
4. 三次握手和四次挥手（A是客户端，B是服务端）
5. 三次握手：
6. 在没有开始连接之前A、B双方都处于关闭状态（CLOSED），A主动打开连接，B被动打开连接，然后B会首先通过TCP服务器进程创建传输控制模块TCB，然后TCP服务器进程进入LISTEN（监听）状态，等待A的连接请求
7. 第一次握手：再发送连接请求报文之前，A首先会通过TCP客户进程创建传输控制模块TCB，然后向B发送连接请求报文（同步标志位SYN=1,序列号seq=x），同时TCP客户进程进入SYN\_SENT（同步发送）状态
8. 第二次握手：B接受到A的连接请求，如果同意连接，则向A发送确认报文（同步标志位SYN=1,确认标志位ACK=1,序列号seq=y,确认号ack=x+1），然后TCP服务器进入SYN\_RCVD（同步收到）状态
9. 第三次握手：A接收到B的确认报文后，A要向B发送确认收到的确认报文（ACK=1,seq=x+1,ack=y+1）,此时ACK报文段可以携带数据，不携带数据则不会消耗序列号，此时TCP连接建立，A进入ESTABLISHED状态
10. 在B收到A的确认后，B也会进入ESTABLISHED状态



1. 四次挥手：
2. A和B都处于ESTABLISHED（已建立连接）状态，A在发送完请求报文之后就会主动发送断开连接请求，即FIN报文（FIN=1，seq=u）告知B数据已经发送完成,同时TCP客户进程进入FIN\_WAIT1状态
3. B接收到A的关闭连接请求后，发送确认接收到关闭请求的报文到A，即（ACK=1,seq=v,ack=u+1），并继续传输尚未完成的数据，此时B进入CLOSE\_WAIT状态，A接收到B的确认报文后继续等待B的FIN报文，并进入FIN\_WAIT2状态
4. B将数据传输完毕后，发送FIN报文到A，即（FIN=1,ACK=1,ack=u+1,seq=w），B进入LAST\_ACK状态
5. 此时A接收到B发送的FIN报文后，A发送确认接收到B的完成报文的报文，即（ACK=1,seq=u+1,ack=w+1），A进入TIME\_WAIT状态,
6. B在接收到A发送的确认报文后进入了CLOSED状态，在2MSL（最大报文段生存时间）内若无数据传输则表示B已经关闭，此时A也进入CLOSED状态



9、内容协商

1. 服务器驱动协商
2. 客户端驱动协商
3. 透明协商：即客户端和服务器分别进行内容协商
4. 内容协商的分类大致为：
5. Accept：接受请求http标头，通告客户端其能够后理解的MIME类型
6. Accept-charset：规定服务器处理表单数据所接受的字符集，用引号包含字符集名称列表，默认是unknow（表单的字符集和包含表单的文档的字符集相同）。字符集一般有：UTF-8-Unicode字符编码、ISO-8859-1拉丁字母，在不匹配时浏览器可以忽略或者区别对待
7. Accept-Encoding：
8. Accept-Language：告知服务器客户端能够处理的自然语言集，q来设置优先级
9. Content-language：
10. MIME：描述消息内容类型的因特网标准，包含文本、视频、音频、图片以及其他程序的二进制数据，即一系列消息类型的集合
11. 权重：用q来表示，若不给则默认权重值是1.0，如：Accept:text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

10、CORS跨域资源共享机制

1. 原因：出于安全考虑，使用额外的HTTP头来告诉浏览器，让运行在一个origin上的web应用可以访问来自不同源服务器上的制定的资源
2. CORS请求分为两种：简单请求和非简单请求
3. 简单请求：只要同时满足一下两大条件，就是简单请求
4. 请求方法三种之一：HEAD\GET\POST
5. HTTP的头信息不超出以下几种字段：Accept、Accept-Language、Content-Language、Last-Event-ID、Content-type：只限于三种，application/x-www-form-urlencoded\multipart/form-data\text-plain
6. withCredentials：设置请求是否可以携带cookie到服务端，即xhr.withCredentials = true，表示可以携带Cookie,当可以发送Cookie时，Access-Control-Allow-Origin就不能设置为通配符“\*”，必须指定为当前发起请求的源，并且Cookie也存在同源策略，所以必须时服务器域名设置的Cookie才可以上传，此时document.cookie才能读取到Cookie的数据
7. 简单请求种，浏览器会在请求头种添加Origin字段，说明本次请求来自哪个源，由服务器判断该源地址是否在允许的范围内，若不在，则返回一个正常的http回应，浏览器发现响应中没有包含Access-Control-Allow-Origin字段则抛出一个错误。并由XHR的onerror回调函数捕获，这种状态无法通过状态码识别，因为HTTP回应的状态码可能是200
8. 除去以上所说的两种情况外，均为非简单请求
9. 非简单请求：例如，PUT\DELETE或Content-type为application/type,
10. 预检请求：在正式开始请求之前会进行一次预检请求，即请求类型为OPTIONS（查询当前请求的源地址是否在服务器的白名单之中，以及可以使用哪些头信息字段，如果在则允许请求，否则则报错），在预检请求的头信息中主要有几个信息：
11. Origin：当前发起请求的源地址
12. Access-Control-Request-Methods：该字段是必须的，即列出浏览器CORS请求会用到那些HTTP方法
13. Access-Control-Request-Headers：该字段是一个以逗号隔开的，浏览器CORS请求会发送到服务器的一些额外头信息字段
14. 正常请求：请求头中浏览器会自动加上Origin字段，同时响应 请求中也会返回对应的Access-Control-Allow-Origin字段，并且 每次请求响应中都会有这两个字段；
15. JSONP：使用目的相同，CORS更加强大，JSONP只支持GET请求，CORS支持所有类型的HTTP请求，JSONP的优势在于支持老旧浏览器，可以对不支持CORS的网站发送请求；
16. 强缓存和弱缓存，协商缓存
17. 强缓存：包括cache-control(http1.1)和Expires(1.0)两种，直接从缓存中读取数据，不再发送请求到服务器，状态码是200（from cache）
18. Expires：HTTP1.0的规范，是一个GMT格式的日期时间，如果当前请求的日期时间在此日期时间之前，则命中缓存，否则就发送请求到服务器端
19. Cache-Control：HTTP1.1的规范，有如下值：
20. max-age：是一个时长，即第一次请求和此值，计算出来的缓存有效时长，然后通过此值与后续的请求日期时间进行比较，如果在此日期时间之前则缓存生效，否则就发送请求到服务器端
21. no-store：禁止浏览器缓存数据，即用户每一次都会发送请求到服务器端，都会下载一次完整的数据进行使用
22. no-cache：不适用本地缓存，需要使用协商缓存，先与服务器确认返回的响应是否被修改，如果之前的请求中存在Etag，则请求的时候需要与服务器端进行验证，若验证通过则使用本地缓存，否则重新下载资源数据
23. public：数据可以被所有用户缓存，包括终端用户和CDN等中间代理商服务器
24. private：数据只可以被终端用户缓存，不允许被CDN等中间代理商服务器缓存
25. 如果Expires和Cache-Control同时存在，则Cache-Control的优先级高于Expires
26. 协商缓存：包括Last-Modified/If-Modified-Since和Etag/If-None-Match，会发送请求到服务器端，由服务器来告知是否使用缓存数据，状态码是304（not modified）。这两对header是成对出现的，即第一次请求数据时，服务器会在响应头中带上某个字段（Last-Modified、Etag）,在下一次请求时浏览器会带上对应的header（If-Modified-Since、If-None-Match），若上一次请求中没有携带上述字段，则浏览器下一次请求中也不会携带对应的字段
27. Last-Modified/If-Modified-Since：两者的值都是GMT格式的日期时间字符串，具体过程就是：判断粒度是秒级的
28. 浏览器第一次请求资源数据时，服务器端会在响应头中添加Last-Modified的header，即该资源的最后一次修改时间
29. 浏览器再次请求资源时，会在请求头中添加If-Modified-Since，此header的值就是上一次响应头中的Last-Modified的值
30. 服务器端在收到浏览器的请求时，会拿If-Modified-Since和服务器端上对应资源的最后修改日期时间的值进行比较，如果命中，则返回304(not modified)，响应头中不再添加Last-Modified的header，且不返回资源数据，若未命中则返回资源数据
31. 浏览器收到304(not -modified)响应时，就会从本地缓存中读取数据
32. 如果协商缓存没有命中，浏览器直接从服务器加载数据，Last-Modified的header会在重新加载数据时更新并携带到浏览器端，那么下次浏览器请求时If-Modified-Since的值就会使上一次的Last-Modified的值
33. Etag/If-None-Match：这两个值是服务器针对每个资源生成的唯一标识字符串，只有资源有变化时该标识才会改变，判断过程跟上述的Last-Modified/If-Modified-Since类似，唯一区别在于每次响应服务器都会添加Etag的header，因为每次和If-None-Match比较时都会重新生成一个Etag来进行比较
34. Etag/If-None-Match的优先级高于Last-Modified,在判断Etag一致的情况下才会继续判断Last-Modified,最后决定是否返回304
35. 用户的操作对缓存使用的影响：

|  |  |  |
| --- | --- | --- |
| 用户操作 | Expires/Cache-Control | Etag/Last-Modified |
| 地址栏回车 | 有效 | 有效 |
| 链接地址跳转 | 有效 | 有效 |
| 新开窗口 | 有效 | 有效 |
| F5刷新 | 无效 | 有效 |
| Ctrl+F5强制刷新 | 无效 | 无效 |

1. 如何重新加载缓存缓存过的资源：通过添加文件版本号来处理
2. 响应状态码
3. 1xx：表示服务器已经接受请求，但需要客户端后续操作
   1. 100：客户端应继续发送请求
   2. 101：（switching protocols），需要切换协议，服务器通过Upgrade响应头字段通知客户端，如WebSocket
4. 2xx：表示响应成功
5. 200：响应成功
6. 201：created, 请求已被实现，并且服务器依据请求需要已经创建了一个新的资源
7. 202：accepted，服务器已经接受请求，但由于性能问题无法及时创建资源
8. 204：no content，服务器成功处理了请求，但不需要返回任何实体内容，204禁止包含任何消息，不应产生文档视图的变化
9. 205：reset content，服务器已经成功处理了请求，但不需要返回任何实体内容，205禁止包含任何消息体，与204不同，此状态的响应要求请求者重置文档视图，例如提交表单后，重置表单的内容以便后续使用
10. 206：partial content,对资源的某一部分进行响应，请求中包含Range字段时，响应需要只返回Range指定的范围即可，且响应中需要包含Content-Range字段，指示返回内容的范围
11. 3xx：表示需要进一步的附加操作以完成请求，这些状态码通常用来重定向，重定向的目标在本次响应的Location字段中指明
12. 301：moved permanently，永久性重定向，表示请求的资源已重新分配URI，以后应该使用本响应返回的若干URI之一进行请求，若该请求不是HEAD/GET，浏览器通常会要求用户确认重定向
13. 302：found，临时性重定向，表示请求的资源临时从不同的URI响应请求，除非指定了Cache-Control或Expires，否则该资源不可被缓存，若不是HEAD/GET，浏览器会要求用户确认，再进行重定向
14. 303：see other，表示由于请求的资源存在着另一个URI，应使用GET方法定向获取请求的资源，该方法的存在主要是为了允许由脚本激活的POST请求输出重定向到一个新的资源，303禁止被缓存
15. 304：not modified，表示客户端发送了一个带条件的GET请求且该请求已被允许，而文档的内容（自上次访问以来或根据携带的条件）并没有改变，304禁止包含消息体。304也是一种缓存机制（协商缓存），服务器会对静态资源文件进行缓存，服务器的响应中通常会包含Etag/Last-Modified，而浏览器下次访问同一URL时，请求响应中会包含If-None-Match/If-Modified-Since（即请求携带的条件）
16. 305：use proxy，使用代理
17. 307：临时重定向，与302Found有着相同的含义
18. 4xx：表示客户端是发生错误的原因所在。除非响应的是一个GET请求，否则服务器就应该返回一个解释当前错误状况的实体
19. 400：bad request，由于包含语法错误，当前请求无法被服务器理解。通常在服务器表单验证失败时返回
20. 401：unauthorized，表示当前请求需要用户验证，响应中会包含一个WWW-Authenticate字段询问用户的授权信息，在下次请求时需要添加Authorization头，HTTP Basic Auth就是这样实现的
21. 403：forbidden，服务器已经理解请求，但是拒绝执行它。通常用于服务器已经知道用户的身份的情况，得知该用户无权进行该操作。
22. 404：not found，请求所希望得到的资源未被在服务器上发现
23. 405：method not allowed，请求行中指定的方法不能被用来请求相应的资源
24. 413：request entity too large，服务器拒绝处理当前的请求，因为该请求的实体数据大小超出了服务器规定的或者能够处理的范围。
25. 414：request URI too large，当URI太长时，服务器可以返回414，当HTTP并未规定URI应当有多长，这时取决于浏览器和服务器的设置，服务器可以根据个人需求设置，但是浏览器是没办法控制的最短是2kb，MSIE和Safari时2kb，Opera是4kb,FireFox是8kb。URI的长度限制会使得HTTP的GET方法使用受限。
26. 5xx：表示服务器在处理请求时有错误或异常状态发生，或者是服务器意识到以当前的软硬件无法完成对当前的请求处理，此时响应中应该给出具体的理由，除非是HEAD请求
27. 500：internal server error，通常是代码出错
28. 501：not implemented，
29. 502：bad gateway（网关），作为网关或者代理工作的服务器尝试执行请求，从上游服务器接收到无效响应。如：HTTP代理翻墙或者nginx反向代理应用
30. 503：service unavailable,
31. 504：gateway time-out，作为网关或者代理工作的服务器尝试执行请求，未能及时从上游服务收到响应
32. 505：HTTP Version not supported,