1. **GET和POST的区别**

http协议定义了很多与服务器交互的方法，最基本的有以下4种:

1. GET：查 获取/查询资源信息
2. POST：改 更新资源信息
3. PUT：增 上传文件，不带验证机制，任何人都可以上传，存在安全性问题，一般不使用
4. DELETE：删 删除文件，不带验证机制，一般不使用

GET和POST的区别

1. GET提交的数据会放在URL之后，以?分割URL和传输数据，参数之间以&相连，如EditPosts.aspx?name=test1&id=123456。  POST方法是把提交的数据放在HTTP包的Body中。
2. GET提交的数据大小有限制（因为浏览器对URL的长度有限制），而POST方法提交的数据没有限制。
3. GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值。
4. GET方式提交数据，会带来安全问题，比如一个登录页面，通过GET方式提交数据时，用户名和密码将出现在URL上，如果页面可以被缓存或者其他人可以访问这台机器，就可以从历史记录获得该用户的账号和密码。
5. **状态码**

作用：状态码用来告诉HTTP客户端，HTTP服务器是否产生了预期的Response。

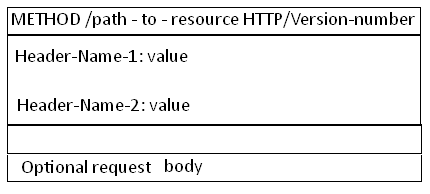
HTTP/1.1中定义了5类状态码， 状态码由三位数字组成，第一个数字定义了响应的类别：

1. 1XX  提示信息 - 表示请求已被成功接收，继续处理
2. 2XX  成功 - 表示请求已被成功接收，理解，接受
3. 3XX  重定向 - 要完成请求必须进行更进一步的处理
4. 4XX  客户端错误 -  请求有语法错误或请求无法实现
5. 5XX  服务器端错误 -   服务器未能实现合法的请求

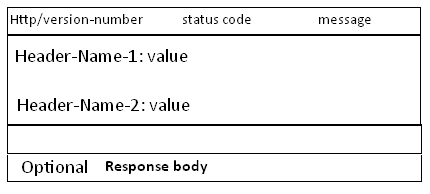
常见的状态码：

1. 200 OK：最常见，这表明该请求被成功地完成，所请求的资源发送回客户端
2. 302 Found：重定向，表明请求的资源已被分配了新的URL，新的URL会在response中的Location中返回，浏览器将会使用新的URL发出新的Request。
3. 304 Not Modified：代表上次的文档已经被缓存了， 还可以继续使用
4. 400 Bad Request：客户端请求与语法错误，不能被服务器所理解
5. 403 Forbidden： 服务器收到请求，但是拒绝提供服务
6. 404 Not Found：请求资源不存在（输错了URL）
7. 500 Internal Server Error 服务器发生了不可预期的错误
8. 503 Server Unavailable 服务器当前不能处理客户端的请求，一段时间后可能恢复正常
9. **request和response的结构**

 Request 消息分为3部分，第一部分叫请求行， 第二部分叫http header, 第三部分是body。header和body之间有个空行， 结构如下图



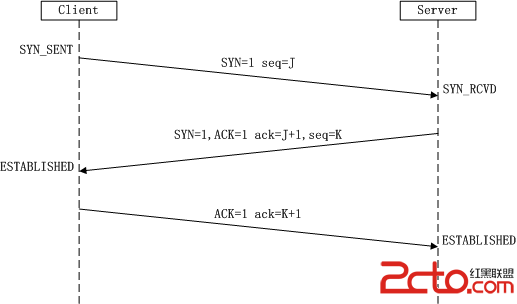
Response结构， 和Request消息的结构基本一样。 同样也分为三部分，第一部分叫request line, 第二部分叫request header，第三部分是body. header和body之间也有个空行，  结构如下图



### （4）从浏览器地址栏输入url到显示页面的步骤(以HTTP为例)

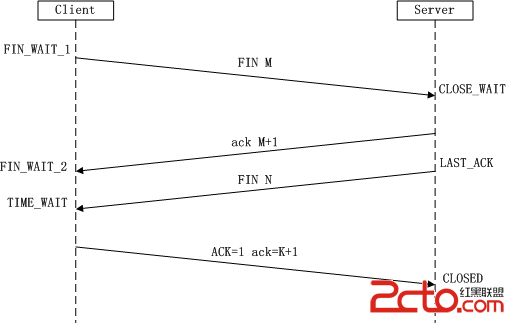
1. 在浏览器地址栏输入URL
2. 浏览器查看**缓存**，如果请求资源在缓存中并且新鲜，跳转到转码步骤
   1. 如果资源未缓存，发起新请求
   2. 如果已缓存，检验是否足够新鲜，足够新鲜直接提供给客户端，否则与服务器进行验证。
   3. 检验新鲜通常有两个HTTP头进行控制Expires和Cache-Control：
      * HTTP1.0提供Expires，值为一个绝对时间表示缓存新鲜日期
      * HTTP1.1增加了Cache-Control: max-age=,值为以秒为单位的最大新鲜时间
3. 浏览器**解析URL**获取协议，主机，端口，path
4. 浏览器**组装一个HTTP（GET）请求报文**
5. 浏览器**获取主机ip地址**，过程如下：
   1. 浏览器缓存
   2. 本机缓存
   3. hosts文件
   4. 路由器缓存
   5. ISP DNS缓存
   6. DNS递归查询（可能存在负载均衡导致每次IP不一样）
6. **打开一个socket与目标IP地址，端口建立TCP链接**，三次握手如下：
   1. 客户端发送一个TCP的**SYN=1，Seq=X**的包到服务器端口
   2. 服务器发回**SYN=1， ACK=X+1， Seq=Y**的响应包
   3. 客户端发送**ACK=Y+1， Seq=Z**
7. TCP链接建立后**发送HTTP请求**
8. 服务器接受请求并解析，将请求转发到服务程序，如虚拟主机使用HTTP Host头部判断请求的服务程序
9. 服务器检查**HTTP请求头是否包含缓存验证信息**如果验证缓存新鲜，返回**304**等对应状态码
10. 处理程序读取完整请求并准备HTTP响应，可能需要查询数据库等操作
11. 服务器将**响应报文通过TCP连接发送回浏览器**
12. 浏览器接收HTTP响应，然后根据情况选择**关闭TCP连接或者保留重用，关闭TCP连接的四次握手如下**：
    1. 主动方发送**Fin=1， Ack=Z， Seq= X**报文
    2. 被动方发送**ACK=X+1， Seq=Z**报文
    3. 被动方发送**Fin=1， ACK=X， Seq=Y**报文
    4. 主动方发送**ACK=Y， Seq=X**报文
13. 浏览器检查响应状态吗：是否为1XX，3XX， 4XX， 5XX，这些情况处理与2XX不同
14. 如果资源可缓存，**进行缓存**
15. 对响应进行**解码**（例如gzip压缩）
16. 根据资源类型决定如何处理（假设资源为HTML文档）
17. **解析HTML文档，构件DOM树，下载资源，构造CSSOM树，执行js脚本**，这些操作没有严格的先后顺序，以下分别解释
18. **构建DOM树**：
    1. **Tokenizing**：根据HTML规范将字符流解析为标记
    2. **Lexing**：词法分析将标记转换为对象并定义属性和规则
    3. **DOM construction**：根据HTML标记关系将对象组成DOM树
19. 解析过程中遇到图片、样式表、js文件，**启动下载**
20. 构建**CSSOM树**：
    1. **Tokenizing**：字符流转换为标记流
    2. **Node**：根据标记创建节点
    3. **CSSOM**：节点创建CSSOM树
21. [**根据DOM树和CSSOM树构建渲染树**](https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction):
    1. 从DOM树的根节点遍历所有**可见节点**，不可见节点包括：1）script,meta这样本身不可见的标签。2)被css隐藏的节点，如display: none
    2. 对每一个可见节点，找到恰当的CSSOM规则并应用
    3. 发布可视节点的内容和计算样式
22. **js解析如下**：
    1. 浏览器创建Document对象并解析HTML，将解析到的元素和文本节点添加到文档中，此时**document.readystate为loading**
    2. HTML解析器遇到**没有async和defer的script时**，将他们添加到文档中，然后执行行内或外部脚本。这些脚本会同步执行，并且在脚本下载和执行时解析器会暂停。这样就可以用document.write()把文本插入到输入流中。**同步脚本经常简单定义函数和注册事件处理程序，他们可以遍历和操作script和他们之前的文档内容**
    3. 当解析器遇到设置了**async**属性的script时，开始下载脚本并继续解析文档。脚本会在它**下载完成后尽快执行**，但是**解析器不会停下来等它下载**。异步脚本**禁止使用document.write()**，它们可以访问自己script和之前的文档元素
    4. 当文档完成解析，document.readState变成interactive
    5. 所有**defer**脚本会**按照在文档出现的顺序执行**，延迟脚本**能访问完整文档树**，禁止使用document.write()
    6. 浏览器**在Document对象上触发DOMContentLoaded事件**
    7. 此时文档完全解析完成，浏览器可能还在等待如图片等内容加载，等这些**内容完成载入并且所有异步脚本完成载入和执行**，document.readState变为complete,window触发load事件
23. **显示页面**（HTML解析过程中会逐步显示页面）
24. **（5）3次握手和4次挥手**

TCP建立连接时进行3次握手，断开时进行4次挥手。



1. 第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server，Client进入SYN\_SENT状态，等待Server确认。
2. 第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求，Server进入SYN\_RCVD状态。
3. 第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，Client和Server进入ESTABLISHED状态，完成三次握手，随后Client与Server之间可以开始传输数据了。

4次挥手：



由于TCP连接时全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭，上图描述的即是如此。  
 （1）第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN\_WAIT\_1状态。  
  （2）第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE\_WAIT状态。  
 （3）第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST\_ACK状态。  
  （4）第四次挥手：Client收到FIN后，Client进入TIME\_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。

1. **为什么建立连接是三次握手，而关闭连接却是四次挥手呢？**  
     这是因为服务端在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。而关闭连接时，当收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据，己方也未必全部数据都发送给对方了，所以己方可以立即close，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK（己方收到了对方关闭的消息）和FIN（己方发送关闭的消息）一般都会分开发送。
2. **TCP和UDP的区别？**

1）TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接；

2）TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付  
3）TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流;UDP是面向报文的  
UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）

4）每一条TCP连接只能点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信  
5）TCP首部开销20字节;UDP的首部开销小，只有8个字节

6）TCP的逻辑通信信道是全双工的可靠信道，UDP则是不可靠信道

（7）http与https

HTTPS = HTTP + SSL

不同之处:

1）HTTP 是不安全的，而 HTTPS 是安全的

2）HTTP 的 URL 以 http:// 开头，而 HTTPS 的 URL 以 https:// 开头

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 工作在传输层

5）HTTP 无需加密，而 HTTPS 对传输的数据进行加密

6）HTTP 无需证书，而 HTTPS 需要认证证书

HTTPS要使客户端与服务器端的通信过程得到安全保证，必须使用的对称加密算法，但是协商对称加密算法的过程，需要使用非对称加密算法来保证安全，然而直接使用非对称加密的过程本身也不安全，会有中间人篡改公钥的可能性，所以客户端与服务器不直接使用公钥，而是使用数字证书签发机构颁发的证书来保证非对称加密过程本身的安全。这样通过这些机制协商出一个对称加密算法，就此双方使用该算法进行加密解密。从而解决了客户端与服务器端之间的通信安全问题。

1. http的header字段

http的header字段包括通用头、请求头、响应头、实体头和扩展头（非标准）这5部分；其中通用头和实体头是在请求header和响应header中都可以出现的。

每个header字段组成：字段名：字段值（大小写无关）

|  |  |
| --- | --- |
| 通用头 | Cache-Control，connection，Date，Via，Pragma，Upgrade，Warning,Transfer-Encoding |
| 请求头 | Accept，User-Agent，Host，**Referer，Range** |
| 响应头 | **Location，Server，Accept-Ranges，Age，** |
| 实体头 | **Content-Type/Range/Length/Location，Last-modified，Allow，** |

1. **XSS攻击**

发生的原理： 因为用户输入的数据变成了代码，改变了源代码；

解决：（1）对用户的输入的"中括号"， “单引号”，“引号” 之类的特殊字符进行编码

（2）值允许用户输入我们期望的数据；

（3）过滤或移除特殊标签

（4）将重要的cookie标记为http only,   这样的话Javascript 中document.cookie语句就不能获取到cookie了

**（9）cookie与session的区别**

1）cookie数据存放在客户的浏览器上，session数据放在服务器上（会在客户端存放一个sessionID）。

2）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。

3）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。

4）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。

5）将登陆信息等重要信息存放为SESSION；其他信息若需要保留，可放在COOKIE中

**（10）HTML，XHTML，html5**

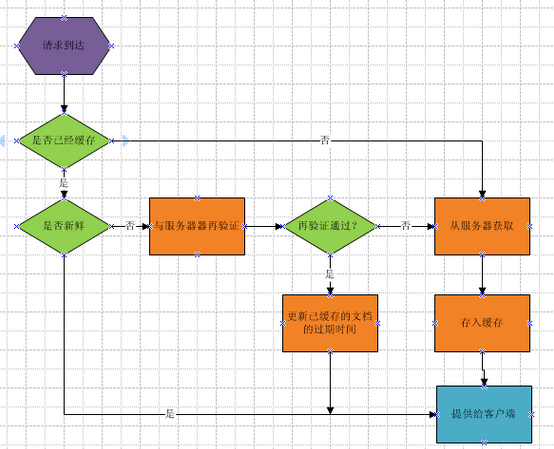
xhtml：所有标签必须被正确嵌套，标签名属性名必须为小写，属性不能为空值，属性值必须用“”等

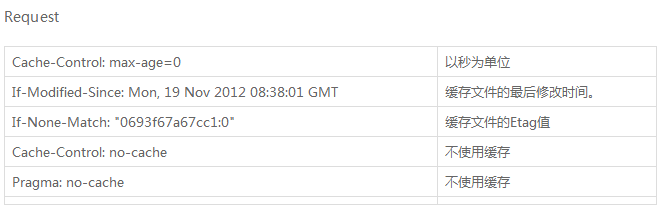
**（11）http缓存**

http中具有缓存功能的是浏览器缓存、以及缓存代理服务器。

浏览器缓存实现：浏览器将最后修改时间发送请求给[web](http://caibaojian.com/c/web)服务器，web服务器收到请求后跟服务器上的文档最后修改的时间对比，如果web服务器上最新文档修改时间小于或者等于浏览器发送过来的，则发送304给浏览器，使用缓存版本。

缓存的好处：1）减少冗余数据传输，节省网络开销；2）提高客户端的加载速度；3）减少服务器负担，提高网站性能。







（12）Etag

ETag是实体标签（Entity Tag）的缩写，可以标识资源的状态。 当资源发送改变时，ETag也随之发生变化。ETag是Web服务端产生的，然后发给浏览器客户端。

为什么使用ETag呢？ 主要是为了解决Last-Modified 无法解决的一些问题。

1. 某些服务器不能精确得到文件的最后修改时间， 这样就无法通过最后修改时间来判断文件是否更新了。

2. 某些文件的修改非常频繁，在秒以下的时间内进行修改. Last-Modified只能精确到秒。

3. 一些文件的最后修改时间改变了，但是内容并未改变。 我们不希望客户端认为这个文件修改了。