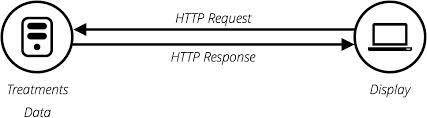
**HTTP协议 处理流程**

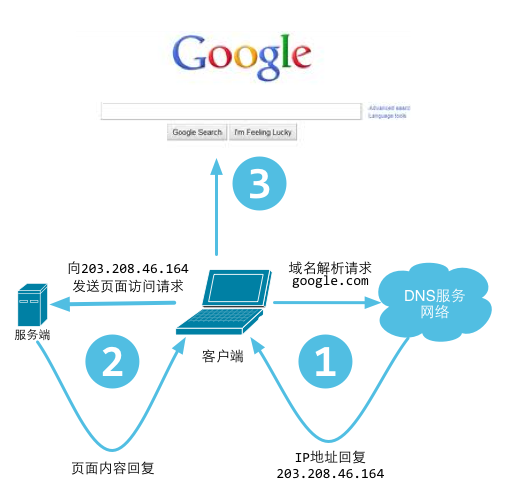
目录结构

* [Web服务器的工作原理可以简单地定义为：](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader0)
* [URL和DNS解析](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader1)
* [DNS工作原理](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader2)
* [DNS解析的整个流程](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader3)
* [HTTP协议详解](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader4)
* [HTTP请求包（浏览器信息）](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader5)
  + [通过fiddler抓包可以看到如下请求信息:](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader6)
  + [看看GET和POST的区别:](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader7)
* [HTTP响应包（服务器信息）](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader8)
* [HTTP协议是无状态的和Connection: keep-alive的区别](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader9)
  + [请求实例](http://www.qixing318.com/article/the-http-protocol-handling-process.html#articleHeader10)



我们平时在浏览网页的时候都是使用浏览器，输入你要的网址后回车，就会显示出我们所想要的内容，看似这个简单的用户操作行为的背后，Web的工作原理是怎样的呢？到底隐藏了些什么呢？

对于传统的上网流程，系统它是这么做的：浏览器本身它是一个客户端，当输入URL地址的时候，浏览器首先会去请求DNS服务器，通过DNS查询获取相应的域名所对应的IP地址，然后通过这个映射的IP地址找到IP对应的服务器，并建立连接，等浏览器发送完HTTP Request（请求）包后，服务器接收到请求包之后才开始处理，返回HTTP Response（响应）包，客户端浏览器收到来自服务器的响应后就开始渲染这个Response包里的主体（body）部分，等收到全部的内容后断开与该服务器之间的连接。



一个Web服务器也被称为HTTP服务器，它通过HTTP协议与客户端通信。这个客户端通常指的是Web浏览器(其实手机端客户端内部也是浏览器实现的)。

Web服务器的工作原理可以简单地定义为：

|  |  |
| --- | --- |
| 1 | 客户机通过TCP/IP协议建立到服务器的TCP连接 |
| 2 | 客户端向服务器发送HTTP协议请求包，请求服务器里的资源文档 |
| 3 | 服务器向客户机发送HTTP协议应答包，如果请求的资源包含有动态语言的内容，那么服务器会调用动态语言的解释引擎负责处理“动态内容”，并将处理得到的数据返回给客户端 |
| 4 | 客户机与服务器断开。由客户端解释HTML文档，在客户端屏幕上渲染图形结果 |

一个简单的HTTP事务就是这样实现的，看起来很复杂，原理其实是挺简单的。需要注意的是客户机与服务器之间的通信是非持久连接的，也就是当服务器发送了应答后就与客户机断开连接，等待下一次请求。

URL和DNS解析

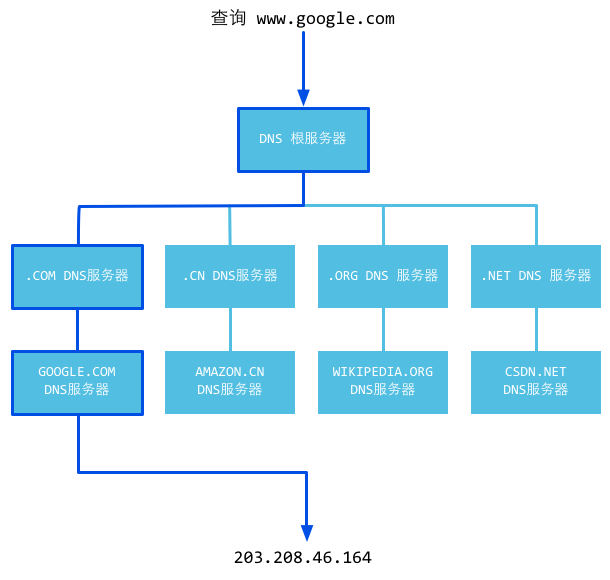
我们浏览网页都是通过URL访问的，那么URL到底是怎么样的呢？

URL(Uniform Resource Locator)是“统一资源定位符”的英文缩写，用于描述一个网络上的资源, 基本格式如下：

scheme://host[:port#]/path/.../[?query-string][#anchor]

|  |  |
| --- | --- |
| scheme | 指定低层使用的协议(例如：http, https, ftp) |
| host | HTTP服务器的IP地址或者域名 |
| port | HTTP服务器的默认端口是80，这种情况下端口号可以省略。如果使用了别的端口，必须指明，例如 http://www.qixing318.com:80/ |
| path | 访问资源的路径 |
| query-string | 发送给http服务器的数据 |
| anchor | 锚 |

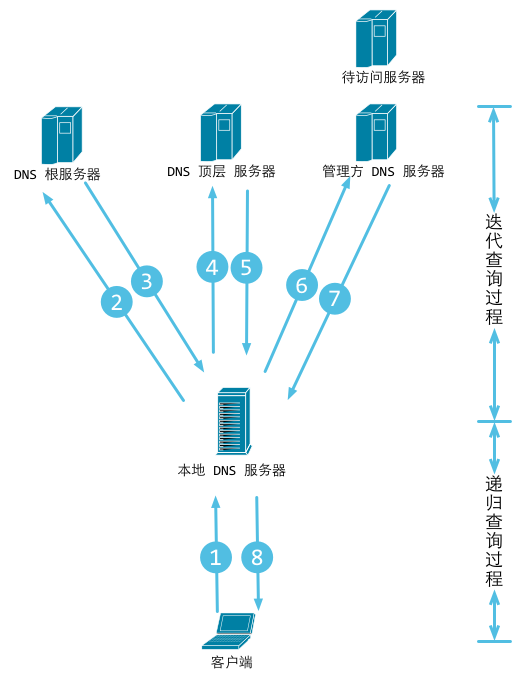
DNS(Domain Name System)是“域名系统”的英文缩写，是一种组织成域层次结构的计算机和网络服务命名系统，它用于TCP/IP网络，它从事将主机名或域名转换为实际IP地址的工作。DNS就是这样的一位“翻译官”，它的基本工作原理可用下图来表示。



DNS工作原理

更详细的DNS解析的过程如下，这个过程有助于我们理解DNS的工作模式

|  |  |
| --- | --- |
| 1 | 在浏览器中输入www.qq.com域名，操作系统会先检查自己本地的hosts文件是否有这个网址映射关系，如果有，就先调用这个IP地址映射，完成域名解析。 |
| 2 | 如果hosts里没有这个域名的映射，则查找本地DNS解析器缓存，是否有这个网址映射关系，如果有，直接返回，完成域名解析。 |
| 3 | 如果hosts与本地DNS解析器缓存都没有相应的网址映射关系，首先会找TCP/IP参数中设置的首选DNS服务器，在此我们叫它本地DNS服务器，此服务器收到查询时，如果要查询的域名，包含在本地配置区域资源中，则返回解析结果给客户机，完成域名解析，此解析具有权威性。 |
| 4 | 如果要查询的域名，不由本地DNS服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个IP地址映射，完成域名解析，此解析不具有权威性。 |
| 5 | 如果本地DNS服务器本地区域文件与缓存解析都失效，则根据本地DNS服务器的设置（是否设置转发器）进行查询，如果未用转发模式，本地DNS就把请求发至 “根DNS服务器”，“根DNS服务器”收到请求后会判断这个域名(.com)是谁来授权管理，并会返回一个负责该顶级域名服务器的一个IP。本地DNS服务器收到IP信息后，将会联系负责.com域的这台服务器。这台负责.com域的服务器收到请求后，如果自己无法解析，它就会找一个管理.com域的下一级DNS服务器地址(qixing318.com)给本地DNS服务器。当本地DNS服务器收到这个地址后，就会找qixing318.com域服务器，重复上面的动作，进行查询，直至找到www.qixing318.com主机。 |
| 6 | 如果用的是转发模式，此DNS服务器就会把请求转发至上一级DNS服务器，由上一级服务器进行解析，上一级服务器如果不能解析，或找根DNS或把转请求转至上上级，以此循环。不管是本地DNS服务器用是是转发，还是根提示，最后都是把结果返回给本地DNS服务器，由此DNS服务器再返回给客户机。 |



DNS解析的整个流程

所谓 递归查询过程 就是 “查询的递交者” 更替, 而 迭代查询过程 则是 “查询的递交者”不变。

举个例子来说，你想知道某个一起上法律课的女孩的电话，并且你偷偷拍了她的照片，回到寝室告诉一个很仗义的哥们儿，这个哥们儿二话没说，拍着胸脯告诉你，甭急，我替你查(此处完成了一次递归查询，即，问询者的角色更替)。然后他拿着照片问了学院大四学长，学长告诉他，这姑娘是xx系的；然后这哥们儿马不停蹄又问了xx系的办公室主任助理同学，助理同学说是xx系yy班的，然后很仗义的哥们儿去xx系yy班的班长那里取到了该女孩儿电话。(此处完成若干次迭代查询，即，问询者角色不变，但反复更替问询对象)最后，他把号码交到了你手里。完成整个查询过程。

通过上面的步骤，我们最后获取的是IP地址，也就是浏览器最后发起请求的时候是基于IP来和服务器做信息交互的。

HTTP协议详解

HTTP协议是Web工作的核心，所以要了解清楚Web的工作方式就需要详细的了解清楚HTTP是怎么样工作的。

HTTP是一种让Web服务器与浏览器(客户端)通过Internet发送与接收数据的协议,它建立在TCP协议之上，一般采用TCP的80端口。它是一个请求、响应协议--客户端发出一个请求，服务器响应这个请求。

在HTTP中，客户端总是通过建立一个连接与发送一个HTTP请求来发起一个事务。服务器不能主动去与客户端联系，也不能给客户端发出一个回调连接。客户端与服务器端都可以提前中断一个连接。例如，当浏览器下载一个文件时，你可以通过点击“停止”键来中断文件的下载，关闭与服务器的HTTP连接。

HTTP协议是无状态的，同一个客户端的这次请求和上次请求是没有对应关系，对HTTP服务器来说，它并不知道这两个请求是否来自同一个客户端。为了解决这个问题， Web程序引入了Cookie机制来维护连接的可持续状态。

HTTP协议是建立在TCP协议之上的，因此TCP攻击一样会影响HTTP的通讯，例如比较常见的一些攻击：SYN Flood是当前最流行的DoS（拒绝服务攻击）与DdoS（分布式拒绝服务攻击）的方式之一，这是一种利用TCP协议缺陷，发送大量伪造的TCP连接请求，从而使得被攻击方资源耗尽（CPU满负荷或内存不足）的攻击方式。

HTTP请求包（浏览器信息）

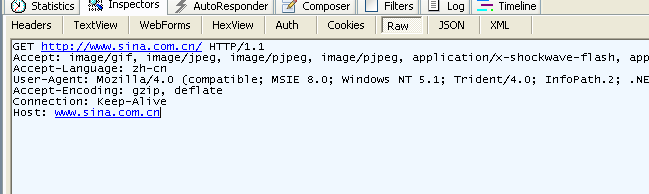
我们先来看看Request包的结构, Request包分为3部分，第一部分叫Request line（请求行）, 第二部分叫Request header（请求头）,第三部分是body（主体）。

header和body之间有个空行，请求包的例子所示:

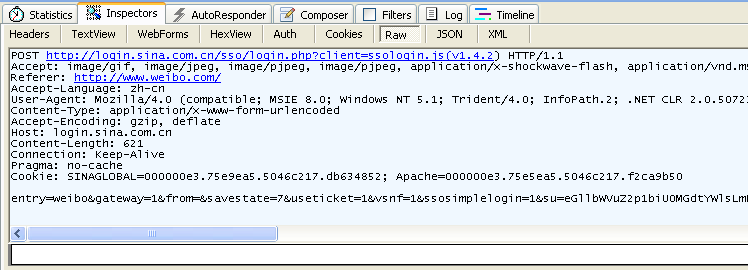
|  |  |
| --- | --- |
| GET /domains/example/ HTTP/1.1 | 请求行: 请求方法 请求URI HTTP协议/协议版本 |
| Host：www.qixing318.com. | 服务端的主机名 |
| User-Agent：Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.4 (KHTML, like Gecko) Chrome/22.0.1229.94 Safari/537.4 | 客户端浏览器信息 |
| Accept：text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8 | 客户端能接收的mine类型 |
| Accept-Encoding：gzip,deflate,sdcn | 是否支持流压缩 |
| Accept-Charset：UTF-8,\*;q=0.5 | 客户端字符编码集 |
| "\r\n\r\n" | 空行,用于分割请求头和消息体 |
| "\r\n\r\n" | 空行,消息体,请求资源参数,例如POST传递的参数 |

HTTP协议定义了很多与服务器交互的请求方法，最基本的有4种，分别是GET,POST,PUT,DELETE。一个URL地址用于描述一个网络上的资源，而HTTP中的GET, POST, PUT, DELETE就对应着对这个资源的查，改，增，删4个操作。我们最常见的就是GET和POST了。GET一般用于获取/查询资源信息，而POST一般用于更新资源信息。

通过fiddler抓包可以看到如下请求信息:



上图为：fiddler抓取的GET信息



上图为：fiddler抓取的POST信息

看看GET和POST的区别:

|  |  |
| --- | --- |
| 1 | 我们可以看到GET请求消息体为空，POST请求带有消息体。 |
| 2 | 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。 |
| 3 | 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。 |
| 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。 |

HTTP响应包（服务器信息）

我们再来看看HTTP的response包，他的结构如下：

|  |  |
| --- | --- |
| HTTP/1.1 200 OK | 状态行 |
| Server: nginx/1.0.8 | 服务器使用的WEB软件名及版本 |
| Date:Date: Tue, 30 Oct 2012 04:14:25 GMT | 发送时间 |
| Content-Type: text/html | 服务器发送信息的类型 |
| Transfer-Encoding: chunked | 表示发送HTTP包是分段发的 |
| Connection: keep-alive | 保持连接状态 |
| Content-Length: 90 | 消息主体内容长度 |
| \r\n | 空行，用来分割消息头和主体 |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"... | 消息体部分 |

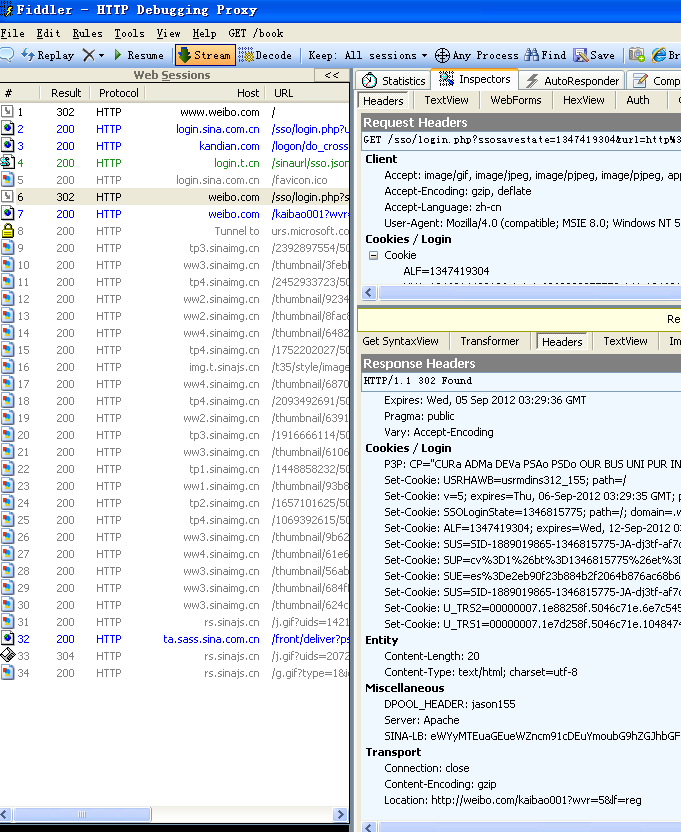
Response包中的第一行叫做状态行，由HTTP协议版本号， 状态码， 状态消息 三部分组成。

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协议中定义了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

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

我们看下面这个图展示了详细的返回信息，左边可以看到有很多的资源返回码，200是常用的，表示正常信息，302表示跳转。response header里面展示了详细的信息。



【上图：访问一次网站的全部请求信息】

HTTP协议是无状态的和Connection: keep-alive的区别

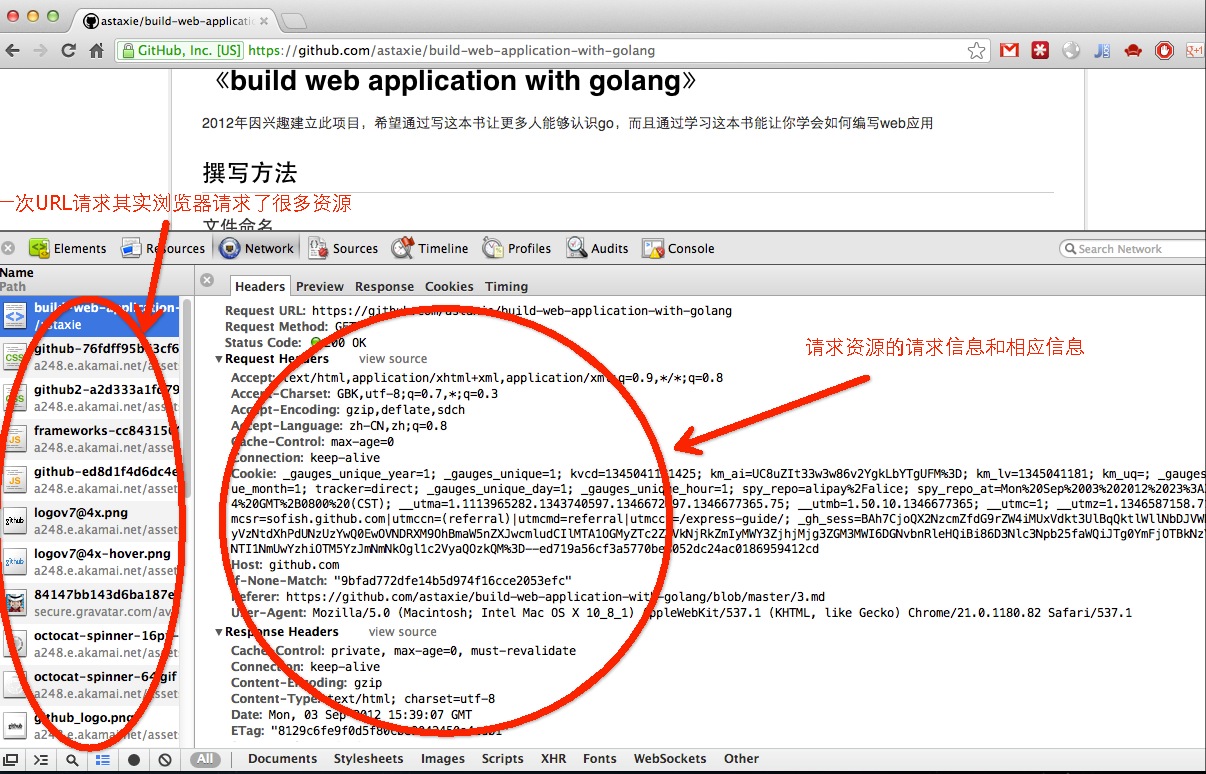
无状态是指协议对于事务处理没有记忆能力，服务器不知道客户端是什么状态。从另一方面讲，打开一个服务器上的网页和你之前打开这个服务器上的网页之间没有任何联系。

HTTP是一个无状态的面向连接的协议，无状态不代表HTTP不能保持TCP连接，更不能代表HTTP使用的是UDP协议（面对无连接）。

从HTTP/1.1起，默认都开启了Keep-Alive保持连接特性，简单地说，当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的TCP连接。

Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同服务器软件（如Apache）中设置这个时间。

请求实例



【上图为：一次请求的request和response】

上面这张图我们可以了解到整个的通讯过程，同时细心的读者是否注意到了一点，一个URL请求但是左边栏里面为什么会有那么多的资源请求(这些都是静态文件，go对于静态文件有专门的处理方式)。

这个就是浏览器的一个功能，第一次请求url，服务器端返回的是html页面，然后浏览器开始渲染HTML：当解析到HTML DOM里面的图片连接，css脚本和js脚本的链接，浏览器就会自动发起一个请求静态资源的HTTP请求，获取相对应的静态资源，然后浏览器就会渲染出来，最终将所有资源整合、渲染，完整展现在我们面前的屏幕上。