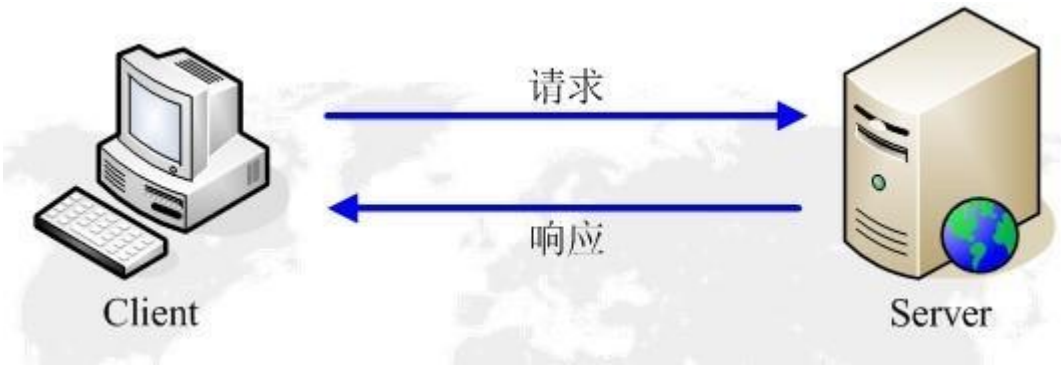
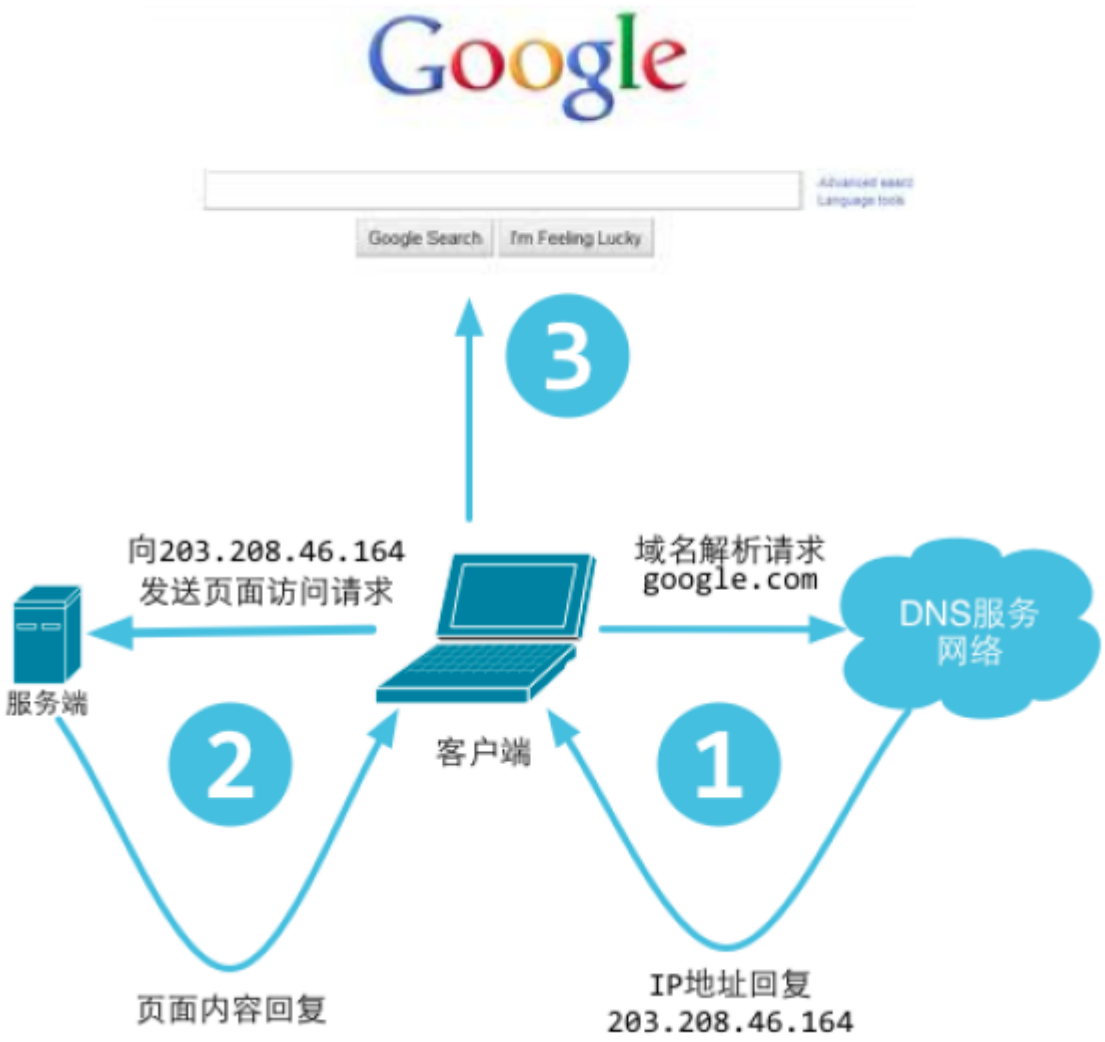
Web 介绍

一、Web工作方式

HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。



我们平时浏览网页的时候，会打开浏览器，输入网址后按下回车键，然后就会显示出你想要浏览的内容。在这个看似简单的用户行为背后，到底隐藏了些什么呢? 对于普通的上网过程，系统其实是这样做的：浏览器本身是一个客户端，当你输入URL的时候，首先浏览器会去请求DNS服务器，通过DNS获取相应的域名对应的IP，然后通过IP地址找到IP对应的服务器后，要求建立TCP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请求包，服务器调用自身服务，返回HTTP Response (响应)包；客户端收到来自服务器的响应后开始渲染这个Response包里的主体(body) ，等收到全部的内容随后断开与该服务器之间的TCP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归纳为:

- 客户端通过TCP/IP协议建立到服务器的TCP连接
- 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
- 服务器向客户端发送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
- 客户端与服务器断开。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。需要注意的是客户端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端断开连接，等待下一次请求。

> 第一次请求url，服务器返回的是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脚本的连接，浏览器会自动发起一个请求静态资源的HTTP请求，获取相应静态资源，然后浏览器会渲染出来，最终将所有资源整合、渲染、完整展现在屏幕上。(网页优化有一向措施是减少HTTP请求次数，把尽量多的css和js资源合并在一起)。

二、URL和DNS解析

2.1 URL

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 (Uni form 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I P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

- path

  访问资源的路径

- query-string

  发送给http服务器的数据

- anchor

  锚

举个例子：

让我们来解析一下下面这一段：

http://mail.163.com/index.html

1、http://:这个是协议，也就是HTTP超文本传输协议，也就是网页在网上传输的协议。

2、mail：这个是服务器名，代表着是一个邮箱服务器，所以是mail.

3、163.com:这个是域名，是用来定位网站的独一无二的名字。

4、mail.163.com：这个是网站名，由服务器名+域名组成。

5、/：这个是根目录，也就是说，通过网站名找到服务器，然后在服务器存放网页的根目录

6、index.html：这个是根目录下的默认网页（当然，163的默认网页是不是这个我不知道，只是大部分的默认网页，都是index.html）

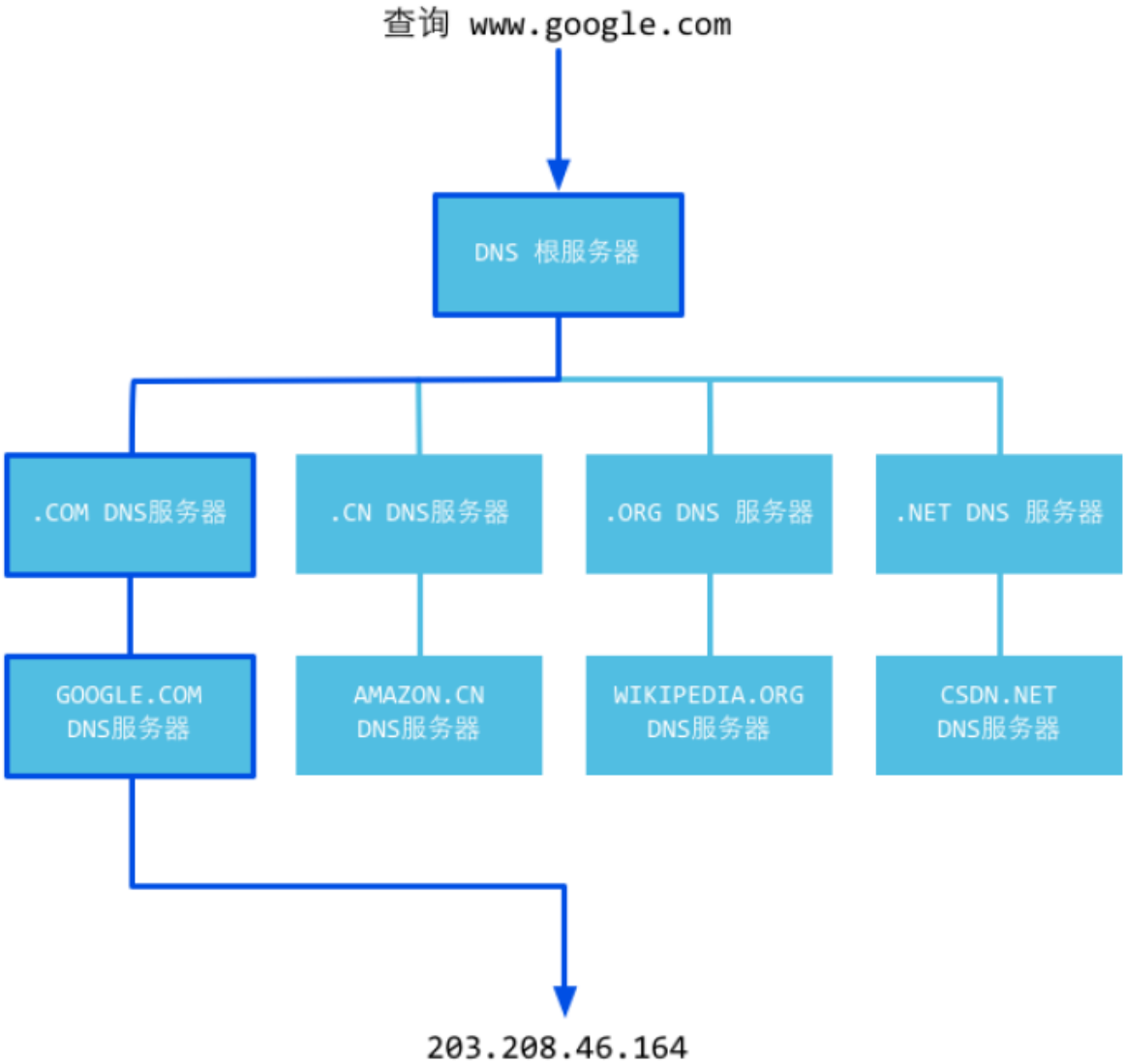
7、http://mail.163.com/index.html:这个叫做URL，统一资源定位符，全球性地址，用于定位网上的资源。

> URI：uniform resource identifier，
>
> 统一资源标识符，用来唯一的标识一个资源。
>
> URL：uniform resource locator，
>
> 统一资源定位器，它是一种具体的URI，即URL可以用来标识一个资源，而且还指明了如何locate这个资源。
>
> URN：uniform resource name，
>
> 统一资源命名，是通过名字来标识资源，比如mailto:java-net@java.sun.com。

> 也就是说，URI是以一种抽象的，高层次概念定义统一资源标识，而URL和URN则是具体的资源标识的方式。URL和URN都是一种URI。

2.2 DNS

DNS (Domain Name System)是“城名系统”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解析过程

1. 浏览器中输入域名，操作系统会先检查自己本地的hosts文件是否有这个网络映射关系，如果有，就先调用这个IP地址映射，完成域名解析。
2. 如果hosts没有域名，查找本地DNS解析器缓存，如果有直接返回，完成域名解析。
3. 如果还没找到，会查找TCP/IP参数中设置的首选DNS服务器，我们叫它本地DNS服务器，此服务收到查询时，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，但该服务已经缓存了地址映射关系，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. 如果上述过程失败，则根据本地DNS服务器的设置进行查询，如果未用转发模式，则把请求发给根服务器，根服务器返回一个负责该顶级服务器的IP，本地DNS服务器收到IP信息后，再连接该IP上的服务器进行解析，如果仍然无法解析，则发送下一级DNS服务器，重复操作，直到找到。
6. 如果是转发模式则把请求转发至上一级DNS服务器，假如仍然不能解析，再转发给上上级。不管是否转发，最后都把结果返回给本地DNS服务器上述一个是迭代查询，一个是递归查询。递归查询的过程是查询者发生了更替，而迭代查询过程，查询者不变。

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。

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-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。

三、HTTP协议

3.1 什么是HTTP协议

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协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写是一个基于TCP/IP通信协议来传递数据，服务器传输超文本到本地浏览器的传送协议，建立在TCP协议之上，一般采用80端口。HTTP协议工作于客户端-服务端架构上。浏览器可作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。它是一个无状态的请求/响应协议。

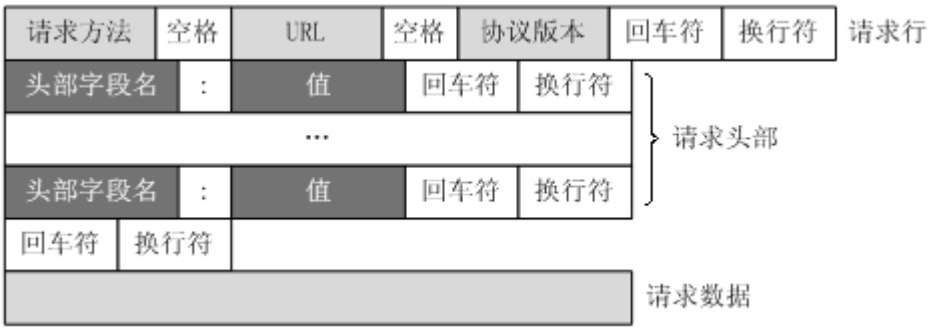
客户端请求消息和服务器响应消息都会包含请求头和请求体。HTTP请求头提供了关于请求或响应，发送实体的信息，如：Content-Type、Content-Length、Date等。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头（server header）用以响应浏览器的请求。

> HTTP是一种让web服务器与客户端通过Internet发送与接受数据的协议，它是一个请求、响应协议，客户端建立连接并发送请求。服务器不能主动去与客户端联系，也不能发送一个回调连接，客户端可提前中断连接。
>
> HTTP请求是无状态的，同一个客户端的每个请求之间没有关联，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。为了解决这个问题引入了cookie机制来维护链接的可持续状态。

3.2 HTTP请求包

我们先来看看Request包的结构：

由 请求行（request line）、请求头部（header）、空行和请求数据四个部分组成。



第一部分：请求行，用来说明请求类型,要访问的资源以及所使用的HTTP版本

GET说明请求类型为GET,[/562f25980001b1b106000338.jpg]为要访问的资源，该行的最后一部分说明使用的是HTTP1.1版本。

第二部分：请求头部，紧接着请求行（即第一行）之后的部分，用来说明服务器要使用的附加信息

从第二行起为请求头部，HOST将指出请求的目的地.User-Agent,服务器端和客户端脚本都能访问它,它是浏览器类型检测逻辑的重要基础.该信息由你的浏览器来定义,并且在每个请求中自动发送等等

第三部分：空行，请求头部后面的空行是必须的

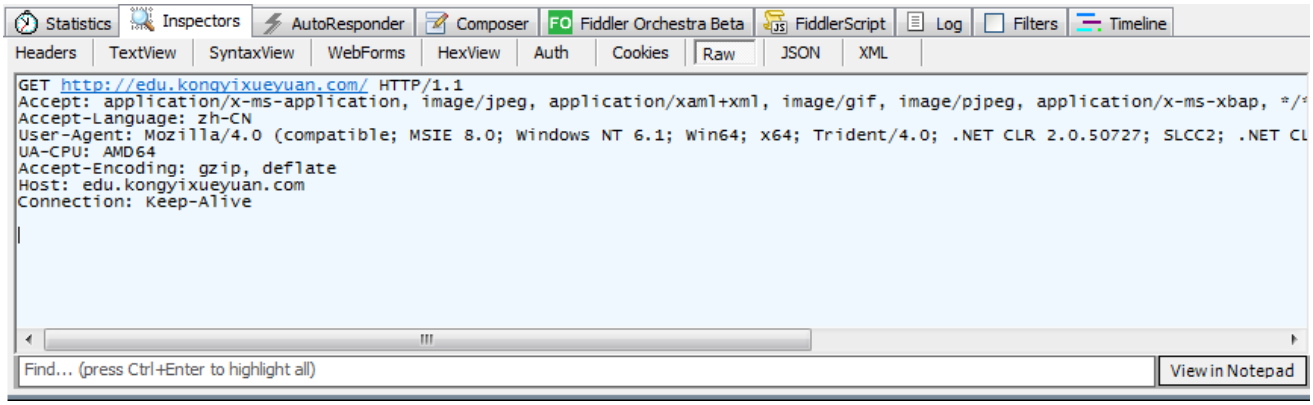
即使第四部分的请求数据为空，也必须有空行。

第四部分：请求数据也叫主体，可以添加任意的其他数据

请求包的例子：

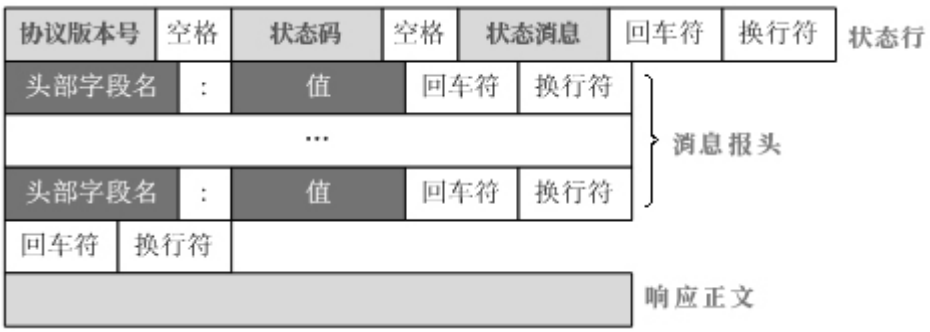
```
GET http://edu.kongyixueyuan.com/ HTTP/1.1            //请求行: 请求方法请求URI HTTP协议/ 协议版本
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg, application/x-ms-xbap, */*                    //客户端能接收的数据格式
Accept-Language: zh-CN
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Win64; x64; Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
UA-CPU: AMD64
Accept-Encoding: gzip, deflate                    //是否支持流压缩
Host: edu.kongyixueyuan.com                    //服务端的主机名
Connection: Keep-Alive
//空行，用于分割请求头和消息体
//消息体,请求资源参数,例如POST传递的参数
```

我们通过fiddler抓包可以看到如下请求信息：



3.3 HTTP响应包

我们再来看看HTTP的response包，也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。



第一部分：状态行，由HTTP协议版本号， 状态码， 状态消息 三部分组成。

第一行为状态行，（HTTP/1.1）表明HTTP版本为1.1版本，状态码为200，状态消息为（ok）

第二部分：消息报头，用来说明客户端要使用的一些附加信息

第二行和第三行为消息报头， Date:生成响应的日期和时间；Content-Type:指定了MIME类型的HTML(text/html),编码类型是UTF-8

第三部分：空行，消息报头后面的空行是必须的

第四部分：响应正文，服务器返回给客户端的文本信息。

空行后面的html部分为响应正文。

响应包的例子：

```
HTTP/1.1 200 OK                                  //状态行
Server: nginx                                    //服务器使用的WEB软件名及版本
Content-Type: text/html; charset=UTF-8           //服务器发送信息的类型
Connection: keep-alive                           //保持连接状态
Set-Cookie: PHPSESSID=mjup58ggbefu7ni9jea7908kub; path=/; HttpOnly
Cache-Control: no-cache
Date: Wed, 14 Nov 2018 08:27:32 GMT              //发送时间
Content-Length: 99324                            //主体内容长度
                                                 //空行用来分割消息头和主体
<!DOCTYPE html>...                               //消息体
```

我们通过fiddler抓包可以看到如下响应信息：

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。(HTTP状态码的英文为HTTP Status Code)

- 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

> 常见状态码：
>
> ```
> 200 OK                  //客户端请求成功
> 400 Bad Request             //客户端请求有语法错误，不能被服务器所理解
> 401 Unauthorized            //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用
> 403 Forbidden               //服务器收到请求，但是拒绝提供服务
> 404 Not Found               //请求资源不存在，eg：输入了错误的URL
> 500 Internal Server Error     //服务器发生不可预期的错误
> 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
> ```

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。

3.4 请求方法

根据HTTP标准，HTTP请求可以使用多种请求方法。 HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。 HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

```
GET       请求指定的页面信息，并返回实体主体。
HEAD      类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
POST      向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
PUT       从客户端向服务器传送的数据取代指定的文档的内容。
DELETE    请求服务器删除指定的页面。
CONNECT   HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
OPTIONS   允许客户端查看服务器的性能。
TRACE     回显服务器收到的请求，主要用于测试或诊断。
```

常用方法: Get\Post\Head

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，**对应着对这个资源的查 ，改 ，增 ，删 4个操作。

另外还有个Head方法. 类似GET方法，只请求页面的首部，不响应页面Body部分，用于获取资源的基本信息，即检查链接的可访问性及资源是否修改。

GET和POST的区别：

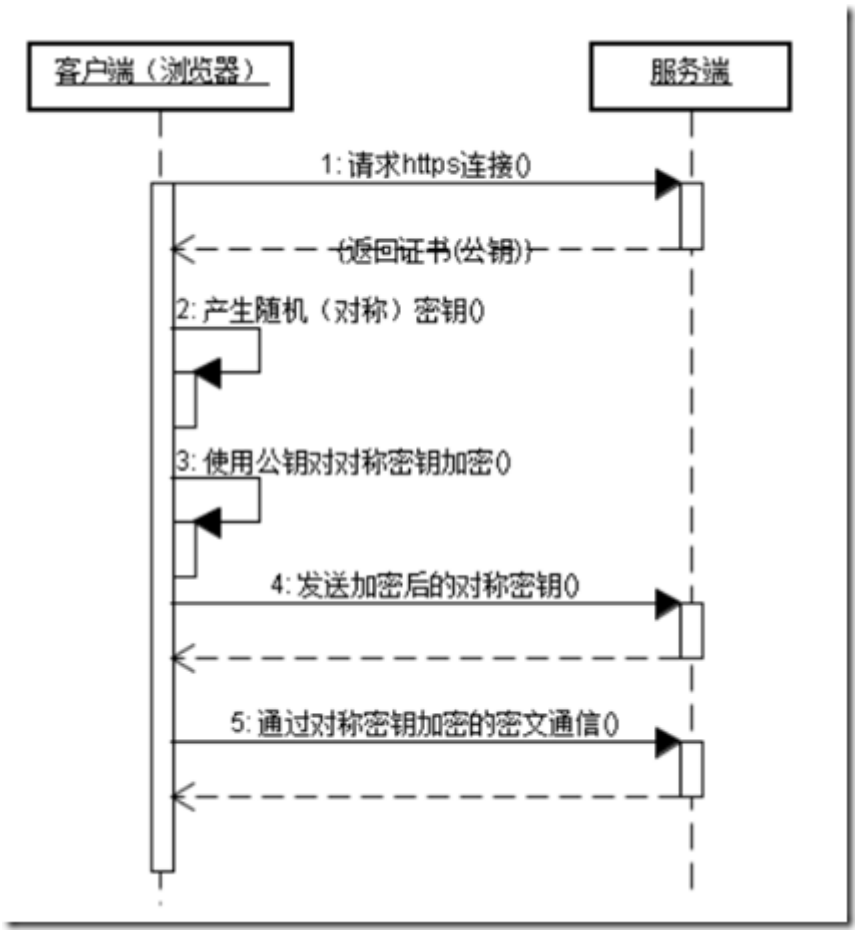
- GET在浏览器回退时是无害的，而POST会再次提交请求。
- GET产生的URL地址可以被Bookmark，而POST不可以。
- GET请求会被浏览器主动cache，而POST不会，除非手动设置。
- GET请求只能进行url编码，而POST支持多种编码方式。
- GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。
- GET请求在URL中传送的参数是有长度限制的，而POST没有。
- 对参数的数据类型，GET只接受ASCII字符，而POST没有限制。
- GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。
- GET参数通过URL传递，POST放在Request body中。

HTTP的底层是TCP/IP。所以GET和POST的底层也是TCP/IP，也就是说，GET/POST都是TCP链接。GET产生一个TCP数据包；POST产生两个TCP数据包。对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据））

四、HTTPS通信原理

HTTPS（Secure Hypertext Transfer Protocol）安全超文本传输协议 它是一个安全通信通道

HTTPS是HTTP over SSL/TLS，HTTP是应用层协议，TCP是传输层协议，在应用层和传输层之间，增加了一个安全套接层SSL。



服务器 用RSA生成公钥和私钥把公钥放在证书里发送给客户端，私钥自己保存客户端首先向一个权威的服务器检查证书的合法性，如果证书合法，客户端产生一段随机数，这个随机数就作为通信的密钥，我们称之为对称密钥，用公钥加密这段随机数，然后发送到服务器服务器用密钥解密获取对称密钥，然后，双方就已对称密钥进行加密解密通信了。

Https的作用

- 内容加密 建立一个信息安全通道，来保证数据传输的安全；
- 身份认证 确认网站的真实性
- 数据完整性 防止内容被第三方冒充或者篡改

Https和Http的区别

- https协议需要到CA申请证书。
- http是超文本传输协议，信息是明文传输；https 则是具有安全性的ssl加密传输协议。
- http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，比http协议安全。