HTTP协议

- HTTP协议（HyperText Transfer Protocol，超文本传输协议）是因特网上应用最为广泛的一种网络传输协议，所有的WWW文件都必须遵守这个标准。

- HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

HTTP请求与响应全过程

- 首先，在浏览器里输入网址
- 浏览器根据域名解析IP地址
- 浏览器与web服务器建立一个 TCP 连接
- 浏览器给Web服务器发送一个HTTP请求
- 服务器的永久重定向响应
- 浏览器跟踪重定向地址
- 服务器“处理”请求
- 服务器发回一个HTML响应
- 释放 TCP 连接

  若connection 模式为close，则服务器主动关闭TCP 连接，客户端被动关闭连接，释放TCP 连接;若connection 模式为keepalive，则该连接会保持一段时间，在该时间内可以继续接收请求;
- 客户端浏览器解析HTML内容
- 浏览器获取嵌入在HTML中的对象

浏览器根据域名解析IP地址

DNS查找过程如下：

1. 浏览器缓存：浏览器会缓存DNS记录一段时间。 但操作系统没有告诉浏览器储存DNS记录的时间，这样不同浏览器会储存个自固定的一个时间（2分钟到30分钟不等）。
2. 系统缓存：如果在浏览器缓存里没有找到需要的域名，浏览器会做一个系统调用（gethostbyname），这样便可获得系统缓存中的记录。
3. 路由器缓存：如果系统缓存也没找到需要的域名，则会向路由器发送查询请求，它一般会有自己的DNS缓存。
4. ISP DNS缓存：如果依然没找到需要的域名，则最后要查的就是ISP缓存DNS的服务器。在这里一般都能找到相应的缓存记录。

域名解析原理：

1. 一个域中的每个主机名与其IP地址的映射关系由这个域的DNS服务器负责管理，例如，”www.it.org”、“ftp.it.org”、“blog.it.org”等主机名都由管理域“it.org”的DNS服务器进行管理，而不能由管理域“org”的DNS服务器进行管理。
2. 每个管理域都必须在其直接父域的DNS服务器上注册该子域的名称和该子域的DNS服务器的IP地址，例如，必须在管理域“org”的DNS服务器注册子域“it.org”和其DNS服务器的IP地址后，域名“it.org”才能真正被外界所认可。
3. 为了方便对顶级域名的统一管理，在顶级域名之上其实还有一个根域名，根域名用点（.）表示，例如，“www.it.org”也可以写为“www.it.org.”，“www.it.org.”中的最后的那个点（.）就表示根域名。 Internet中的根域名由InterNIC（国际互联网络信息中心）集中管理，顶级域名和其下的域名则由拥有该域名的组织、公司和个人自己管理。

HTTP的TCP连接方式

- **HTTP 无状态性**

  HTTP 协议是无状态的(stateless)。也就是说，同一个客户端第二次访问同一个服务器上的页面时，服务器无法知道这个客户端曾经访问过，服务器也无法分辨不同的客户端。HTTP 的无状态特性简化了服务器的设计，使服务器更容易支持大量并发的HTTP 请求。

- **HTTP 持久连接**

  HTTP1.0 使用的是非持久连接，主要缺点是客户端必须为每一个待请求的对象建立并维护一个新的连接，即每请求一个文档就要有两倍RTT 的开销。因为同一个页面可能存在多个对象，所以非持久连接可能使一个页面的下载变得十分缓慢，而且这种短连接增加了网络传输的负担。HTTP1.1 使用持久连接keepalive，所谓持久连接，就是服务器在发送响应后仍然在一段时间内保持这条连接，允许在同一个连接中存在多次数据请求和响应，即在持久连接情况下，服务器在发送完响应后并不关闭TCP 连接，而客户端可以通过这个连接继续请求其他对象。

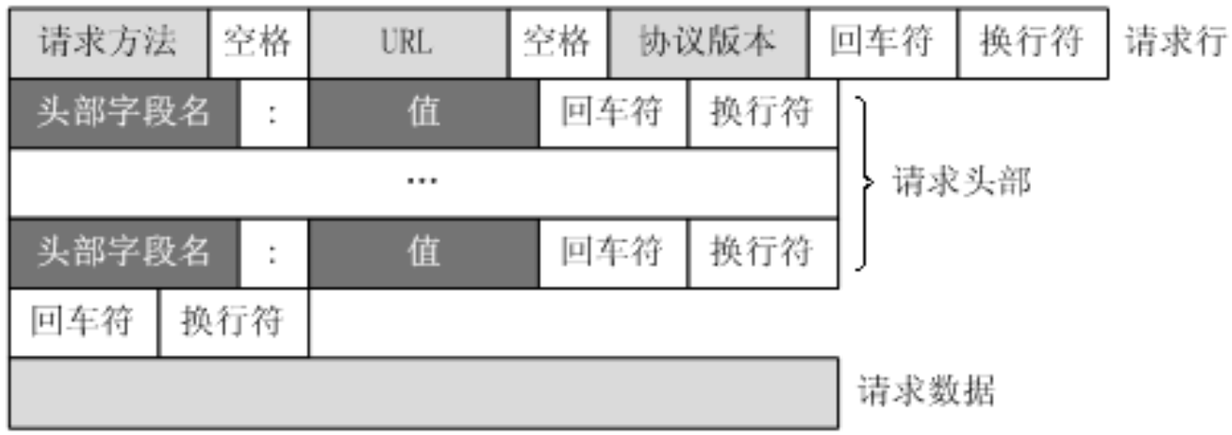
- **HTTP/1.1 协议的持久连接有两种方式：**

  非流水线方式：客户在收到前一个响应后才能发出下一个请求;

  流水线方式：客户在收到 HTTP 的响应报文之前就能接着发送新的请求报文;

HTTP之请求消息

客户端发送一个HTTP请求到服务器的请求消息包括以下格式:
请求行（request line）、请求头部（header）、空行和请求数据四个部分组成。



GET请求的例子:

```
GET /m.jpeg HTTP/1.1
Host: 127.0.0.1:8888
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:42.0) Gecko/20100101 Firefox/42.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

GET请求的例子：

```
GET /m.jpeg HTTP/1.1
Host: 127.0.0.1:8888
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:42.0) Gecko/20100101
Firefox/42.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

- **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,[/m.jpeg]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。**

  这个例子的请求数据为空。

请求行

由**请求方法**、**URL**和**HTTP协议版本**3个字段组成，它们用空格分隔。

例如，GET /index.html HTTP/1.1。

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。

常用的几种请求方法

GET：当客户端要从服务器中读取文档时，当点击网页上的链接或者通过在浏览器的地址栏输入网址来浏览网页的，使用的都是GET方式。GET方法要求服务器将URL定位的资源放在响应报文的数据部分，回送给客户端。使用GET方法时，请求参数和对应的值附加在URL后面，利用一个问号（“?”）代表URL的结尾与请求参数的开始，传递参数长度受限制。例如，/index.jsp?id=100&op=bind。通过GET方式传递的数据直接放在在地址中，所以GET方式的请求一般不包含”请求内容”部分，请求数据以地址的形式表现在请求行。地址中”?”之后的部分就是通过GET发送的请求数据，我们可以在地址栏中清楚的看到，各个数据之间用”&”符号隔开。显然这种方式不适合传送私密数据。另外，由于不同的浏览器对地址的字符限制也有所不同，一般最多只能识别1024个字符，所以如果需要传送大量数据的时候，也不适合使用GET方式。如果数据是英文字母/数字，原样发送，如果是空格，转换为+，如果是中文/其他字符，则直接把字符串用BASE64加密，得出如：%E4%BD%A0%E5%A5%BD，其中％XX中的XX为该符号以16进制表示的ASCII。

POST：允许客户端给服务器提供信息较多。POST方法将请求参数封装在HTTP请求数据中，以名称/值的形式出现，可以传输大量数据，这样POST方式对传送的数据大小没有限制，而且也不会显示在URL中。POST方式请求行中不包含数据字符串，这些数据保存在“请求内容”部分，各数据之间也是使用”&“符号隔开。POST方式大多用于页面的表单中。因为POST也能完成GET的功能，因此多数人在设计表单的时候一律都使用POST方式，其实这是一个误区。GET方式也有自己的特点和优势，我们应该根据不同的情况来选择是使用GET还是使用POST。

HEAD：就像GET，只不过服务端接受到HEAD请求后只返回响应头，而不会发送响应内容。当我们只需要查看某个页面的状态的时候，使用HEAD是非常高效的，因为在传输的过程中省去了页面内容。

请求头部

由关键字/值对组成，每行一对，关键字和值用英文冒号":"分隔。请求头部通知服务器有关于客户端请求的信息

典型的请求头有：

User-Agent：产生请求的浏览器类型。

Accept：客户端可识别的内容类型列表。星号“*”用于按范围将类型分组，用“*/*”指示可接受全部类型，用“type/*”指示可接受 type 类型的所有子类型。

Host：要请求的主机名，允许多个域名同处一个IP地址，即虚拟主机。

Accept-Language：客户端可接受的自然语言。

Accept-Encoding：客户端可接受的编码压缩格式。

Accept-Charset：可接受的应答的字符集。

connection：连接方式(close 或 keepalive)。

Cookie：存储于客户端扩展字段，向同一域名的服务端发送属于该域的cookie。

空行

最后一个请求头部之后是一个空行，发送回车符和换行符(\r\n)，通知服务器以下不再有请求头部。

请求数据

请求数据不在GET方法中使用，而在POST方法中使用。POST方法适用于需要客户填写表单的场合。与请求数据相关的最常使用的请求头部是Content-Type和Content-Length。

HTTP之响应消息

一般情况下，服务器接收并处理客户端发过来的请求后会返回一个HTTP的响应消息。

HTTP响应也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。

响应的例子：

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 5

Hello
```

响应的例子：

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 5

Hello
```

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，

Content-Type:指定了MIME类型的HTML(text/html)

Content-Length:指定了传输数据的长度

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Hello部分为响应正文。

状态行

由**HTTP协议版本**、服务器返回的**响应状态码**和**响应状态码的文本描述**组成。

例如：HTTP/1.1 200 OK

状态代码

有三位数字组成，第一个数字定义了响应的类别，共分五种类别:

1xx：指示信息--表示请求已接收，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操作

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

响应头部

由关键字/值对组成，每行一对，关键字和值用英文冒号”:“分隔。
典型的响应头有：

Location：用于重定向接受者到一个新的位置。例如：客户端所请求的页面已不存在原先的位置，为了让客户端重定向到这个页面新的位置，服务器端可以发回Location响应报头后使用重定向语句，让客户端去访问新的域名所对应的服务器上的资源

Server：包含了服务器用来处理请求的软件信息及其版本。它和 User-Agent 请求报头域是相对应的，前者发送服务器端软件的信息，后者发送客户端软件(浏览器)和操作系统的信息

Vary：指示不可缓存的请求头列表

Connection：连接方式

对于请求来说：close(告诉 WEB 服务器或者代理服务器，在完成本次请求的响应后，断开连接，不等待本次连接的后续请求了)。keepalive(告诉WEB服务器或者代理服务器，在完成本次请求的响应后，保持连接，等待本次连接的后续请求);

对于响应来说：close(连接已经关闭); keepalive(连接保持着，在等待本次连接的后续请求); Keep-Alive：如果浏览器请求保持连接，则该头部表明希望WEB 服务器保持连接多长时间(秒);例如：Keep-Alive：300;

WWW-Authenticate：必须被包含在401 (未授权的)响应消息中，这个报头域和前面讲到的Authorization 请求报头域是相关的，当客户端收到 401 响应消息，就要决定是否请求服务器对其进行验证。如果要求服务器对其进行验证，就可以发送一个包含了Authorization 报头域的请求

空行

最后一个响应头部之后是一个空行，发送回车符和换行符，通知浏览器以下不再有响应头部。

响应数据

服务器返回给客户端的文本信息。