# 一、http基础

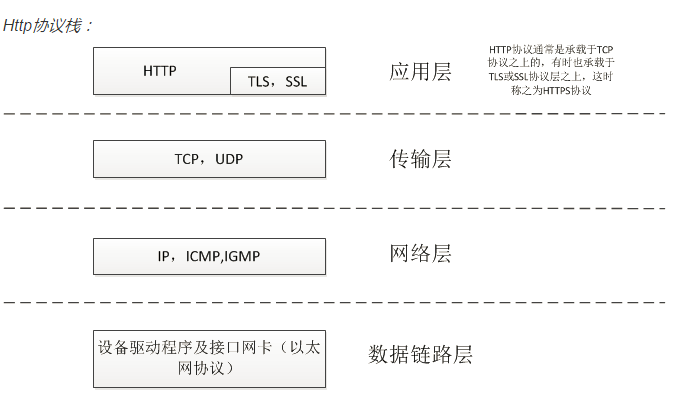
http协议即超文本传输协议，是无状态的、应用层协议。

## 理解：http协议是无状态的

无状态是指的是同一个会话的连续2次请求互相不了解。当客户端与服务器端动态交互的web应用出现后，http的无状态的特点严重影响了这些应用的效果，因此，两种用于保持http连接状态的技术就应运而生：cookie和session.

## 理解：http为什么设计成无状态的？

在一个互联网应用中，服务器需要处理大量的客户端请求，http协议应尽可能少的减少资源占用，所以设计为无状态.



我们现在用到的版本是**HTTP/1.1**，它比1.0版本添加了更多特性。其中比较重要的特性有：

\* 支持持久连接

\* 支持消息切分成块传输

\* 更加丰富的cache特性

\* 带宽优化及网络连接的使用

\* 错误通知的管理

\* 互联网地址的维护安全性及完整性

# 二、请求方式

对于服务器发起的url请求，需要告诉服务器请求的目的是什么，http1.1共定义了8种请求方式:

GET: 向指定的资源发出“显示”请求。使用GET方法应该只用在读取数据，而不应当被用于产生“副作用”的操作中，例如在Web Application中。其中一个原因是GET可能会被网络蜘蛛等随意访问。

POST: 向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。数据被包含在请求本文中。这个请求可能会创建新的资源或修改现有资源，或二者皆有。

PUT: 向指定资源位置上传其最新内容。

DELETE: 请求服务器删除Request-URI所标识的资源。

OPTIONS: 这个方法可使服务器传回该资源所支持的所有HTTP请求方法。用’\*’来代替资源名称，向Web服务器发送OPTIONS请求，可以测试服务器功能是否正常运作。

HEAD: 与GET方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回资源的本文部分。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”（元信息或称元数据）。

TRACE: 回显服务器收到的请求，主要用于测试或诊断。

CONNECT: HTTP/1.1协议中预留给能够将连接改为渠道方式的代理服务器。通常用于SSL加密服务器的链接（经由非加密的HTTP代理服务器）。

Method名称是区分大小写的。当某个请求所针对的资源不支持对应的请求方法的时候，服务器应当返回状态码405（Method Not Allowed），当服务器不认识或者不支持对应的请求方法的时候，应当返回状态码501（Not Implemented）。

GET(获取)、POST(提交)、PUT(修改)、DELETE(删除)，其中我们常用的是GET和POST。

## 理解：GET和POST的异同？

其实GET是为了获取数据的，POST是为了提交数据的

## 常见误区

参见：http://www.nowamagic.net/librarys/veda/detail/1919

1. post比get提交更多的数据？

这个说法是错误的，http协议并没有规定这一项，只不过不同的浏览器厂商对其有一定的限制罢了，之所以有的浏览器限制get的长度，是怕无限制时被有人恶意乱搞，get包含的数据太大对服务器也会造成压力。

1. post比get更安全？

安全也是相对的，他们本来都不安全。Post请求，通过抓包，我们任然能得到其中的数据。

# 三、状态码

状态码是一种非常重要的响应信息，客户端通过状态码就可以知道服务器做出了何种响应。http1.1协议定义了5种类型的状态码：

1xx: 表示请求已被成功接收，告诉客户端可以继续发送下一个请求了，若如果已发送完毕可以忽略它。

2xx:成功

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，语法有误或者请求无法实现

5xx:服务器端错误，服务器未能实现合法的请求。

常见的状态码：

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

最常见的Response处理成功之后返回的状态码：

HTTP/1.1 200 OK （CRLF）

# 四、http消息结构

## 大致结构

HTTP消息分为Request和Response两种消息，Http协议对这两种消息定义了如下结构：

message = <start-line>

\*(<message-header>)

CRLF

[<message-body>]

<start-line> = Request-Line | Status-Line

<message-header> = Field-Name ':' Field-Value

从上面定义我们可以看到，Http的Request和Response消息都是由三部分组成：

1. start-line 开始行

2. header 消息头

3. body 消息体

对于start-line，又分为：

1. Request-Line : 'METHOD/path-to-resource http-version'

2. Response-Line : 'http-version status-code message'

对于Headers则有如下几种：

1. general headers

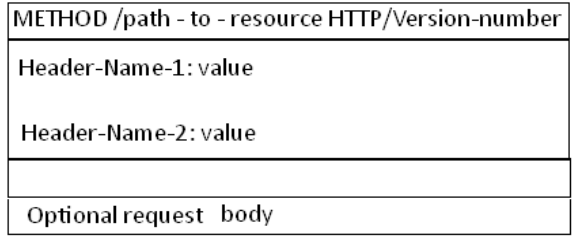
2. entity headers

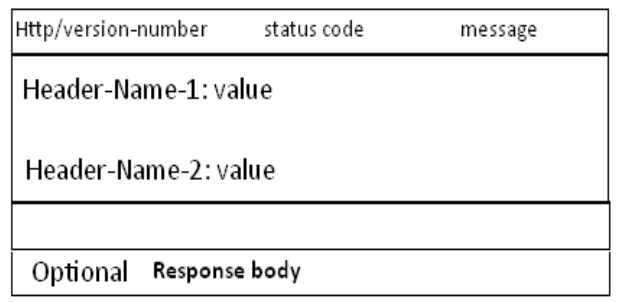
3. request or response headers

a. request specific headers.

b. response specific headers.

下面我们再用两幅图具体看一下Request和Response的消息格式：

Request消息格式定义:   


Response消息格式定义:   


## Headers部分

通过上面介绍我们已经Http消息的Headers共分为三种，分别是General Headers、Entity Headers、Request/Response Headers。

### General headers

我把被Request和Response共享的Headers成为General Headers，具体有：

general-header = Cache-Control

| Connection

| Date

| Pragma

| Trailer

| Transfer-Encoding

| Upgrade

| Via

| Warning

* Cache -Control指定请求和响应遵循的缓存机制。
* Connection 允许客户端和服务器指定与请求/响应连接有关的选项
* Date 提供日期和时间标志,说明报文是什么时间创建的
* Pragma头域用来包含实现特定的指令，最常用的是Pragma:no-cache
* Trailer 如果报文采用了分块传输编码(chunked transfer encoding) 方式,就可以用这个首部列出位于报文拖挂(trailer)部分的首部集合
* Transfer-Encoding 告知接收端为了保证报文的可靠传输,对报文采用了什么编码方式
* Upgrade 给出了发送端可能想要”升级”使用的新版本和协议
* Via 显示了报文经过的中间节点(代理,网嘎un)

### Entity headers

EntityHeaders主要用来描述消息体（message body）的一些元信息，具体有：

entity-header = Allow

| Content-Encoding

| Content-Language

| Content-Length

| Content-Location

| Content-MD5

| Content-Range

| Content-Type

| Expires

| Last-Modified

其中，以Content为前缀的Headers主要描述了消息体的结构、大小、编码等信息，Expires描述了Entity的过期时间，Last-Modified描述了消息的最后修改时间。

# 五、Request实例

## 请求行

Request-Line是Request消息体的第一部分，其具体定义如下：

Request-Line = Method SP URI SP HTTP-Version CRLF

Method = "OPTIONS"

| "HEAD"

| "GET"

| "POST"

| "PUT"

| "DELETE"

| "TRACE"

其中sp代码字段的分隔符，HTTP-Version一般就是"http/1.1"，后面紧接着是一个换行。

## 请求头

在Request-Line后面紧跟着的就是Headers。我们在上面已经介绍了General Headers和Entity Headers，下面便是Request Headers定义：

request-header = Accept

| Accept-Charset

| Accept-Encoding

| Accept-Language

| Authorization

| Expect

| From

| Host

| If-Match

| If-Modified-Since

| If-None-Match

| If-Range

| If-Unmodified-Since

| Max-Forwards

| Proxy-Authorization

| Range

| Referer

| TE

| User-Agent

Request Headers扮演的角色其实就是一个Request消息的调节器。需要注意的是若一个headers名称不在上面列表中，则默认当做Entity Headers的字段。

前缀为Accept 的headers定义了客户端可以接受的媒介类型、语言和字符集等。From, Host, Referer 和User-Agent 详细定义了客户端如何初始化Request。前缀为If 的headers规定了服务器只能返回符合这些描述的资源，若不符合, 则会返回304 Not Modified。

## 请求体

若Request-Line中的Method为GET，请求中不包含消息体，若为POST，则会包含消息体。

## 请求实例：

GET /articles/http-basics HTTP/1.1

Host: www.articles.com

Connection: keep-alive

Cache-Control: no-cache

Pragma: no-cache

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

# 六、Response消息体

Response消息格式和Request类似，也分为三部分：Response-Line、Response Headers、Response Body。

## 响应行

Response-Line具体定义如下：

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

* HTTP-Version字段值一般为HTTP/1.1
* Status-Code前面已经讨论过了
* Reason-Phrase 是对status code的具体描述

一个最常见的Response响应为:

HTTP/1.1 200 OK

## 响应头

下面是response-header的定义：

response-header = Accept-Ranges

| Age

| ETag

| Location

| Proxy-Authenticate

| Retry-After

| Server

| Vary

| WWW-Authenticate

* Age 表示消息自server生成到现在的时长，单位是秒
* ETag 是对Entity进行MD5 hash运算的值，用来检测更改
* Location 被重定向的URL
* Server 服务器标识

## 响应体

具体的html数据