[HTTP][https://www.jmarshall.com/easy/http/#http1.1s9]

[HTTP][https://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177]为超文本传输协议([hypertext transfer protocol][https://www.w3.org/Protocols/rfc2616/rfc2616.html])。stateless，分布式系统之间的用于交流的应用层面的协议，是现代网页的基础。

HTTP Basics

HTTP允许不同的主机(hosts)和用户(clients)间交流，同时支持多种网络配置 。

To make this possible, it assumes very little about a particular system, and does not keep state between different message exchanges

为保证HTTP成为一个stateless协议。该通讯常采用TCP/IP接口(port)，同时任何可信赖的传输都可采用。默认TCP/IP接口(port)为80，同时也可采用其他接口。

custom headers can also be created and sent by the client

交流发生在主机和用户之间，通过request/response 对。用户初始一个HTTP request信息，同时通过一个HTTP response信息作为回馈。

当前的协议版本为HTTP/1.1, 该协议相对于1.0版本增添了少量特色。其中最重要的包含：persistent connections, chunked transfer-coding and fine-grained caching headers。

URLs

网络交流的核心就是request message，通过Uniform Resource Locators(URLs)发送。URLs拥有简单结构，包含以下部分：

一般为http，对于secure communications可以为https。默认接口(port)为80。resource path为本地道服务器资源的路径。

Werbs, there are also web debugging proxies, like fiddler on Windows and charles proxy for OSX

URLs代表了想要交流的主机身份，但该行为需要通过HTTP verbs来执行。同时用户希望主机执行多个行为。这些request verbs包含：

GET: 取回存在的资源, URLs包含了包含了服务器用于定位和返回资源的所需的所有必要信息

POST: 创建新的资源, POST requests常携带用于构建新资源的数据, payload

PUT: 更新现存资源, payload可能包含用于更新的资源

DELETE: 删除现存资源

以下为一些较少使用的HTTP verbs：

HEAD: 类似于GET, 但是不含信息主体. 常用于回溯特殊资源的服务器headers, 一般用于查看资源是否通过timestamps改变

TRACE: used to retrieve the hops that a request takes to round trip from the server. Each intermediate proxy or gateway would inject its IP or DNS name into the `via` header field. This can be used for diagnostic purposes.

OPTIONS: 用于回溯服务器承载力. 在客户端侧, 可用于根据服务器所支持修改request

Status Codes

通过URLs和verbs，用户可初始对服务器的requst。作为回馈，服务器响应statues和信息payloads。status code告知用户服务器响应状态。HTTP定义了不同响应类型的明确数字范围:

1xx: Informational Message, 服务器发送, Expect: 100-continue信息, 告诉用户持续发送剩余的request或者在已发送情况下选择忽略

2xx: Successful, 告知用户request成功, 最常用为200 ok

3xx: Redirection, 需要用户选择额外的行为, 最常用的是跳到其他不同的URL来获取资源

4xx: Client Error, 当服务器认为用户存在故障, 无论是通过一个无效的资源还是发送了一个错误的request, 404 Not Found

5xx: Server Error, 表明服务器在处理request时失败, 503 Service Unavailable

Request and Response Message Formats

HTTP声明request或response信息遵循以下结构：

```
message = <start-line>
          *(<message-header>)
          CRLF
          [<message-body>]

<start-line> = Request-Line | Status-Line
<message-header> = Field-Name ':' Field-Value
```

在message headers和message body间强制增加新的行。该信息可包含一个或多个headers，这些信息广泛分类为：

general headers, 应用于request和response message

request specific headers

response specific headers

entity headers

message body可包含完整的数据，也可在Transfer-Encoding: chunked使用时包含一块一块的数据。

General Headers

以下为request和response messages所支持的一般headers

```
general-header = Cache-Control
               | Connection
               | Date
               | Pragma
               | Trailer
               | Transfer-Encoding
               | Upgrade
               | Via
               | Warning
```

- Via header用于所有intermittent proxies和gateways追踪和更新
- Pragma为custom header，可用于包含implementation-specific headers，最常用的pragma-directive为Pragma: no-cache
- Date header用于timestamp the request/response message
- Upgrade 用于替换协议，允许顺利转到新协议上
- Transfer-Encoding 用于将response分更小部分(Transder-Encoding: chunked)

Entity headers可被request和response包含，提供包含content(aka Message Body or Entity)meta-information:

```
entity-header  = Allow
               | Content-Encoding
               | Content-Language
               | Content-Length
               | Content-Location
               | Content-MD5
               | Content-Range
               | Content-Type
               | Expires
               | Last-Modified
```

- 所有Content-开头的headers提供了关于结构的信息，编码了信息主体的大小。
- Expires header指明了entity 到期的timestamp。
- Last-Modified header指明了entity最后一次修改的timestamp

Custom headers can also be created and sent by the client; they will be treated as entity headers by the HTTP protocol.

Request Format拥有类似的结构：

```
Request-Line = Method SP URI SP HTTP-Version CRLF
Method = "OPTIONS"
       | "HEAD"
       | "GET"
       | "POST"
       | "PUT"
       | "DELETE"
       | "TRACE"
```

典型的request message可能看起来类似：

```
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=(
```

Response Format类似于request message, 除了status line和headers。status line具有以下结构：

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

- HTTP-version 为HTTP/1.1
- Status-code
- SP, space separator between the tokens
- Reason-Phrase为人类可识别的状态码

例如：

```
HTTP/1.1 200 OK
```
