HTTP 协议入门

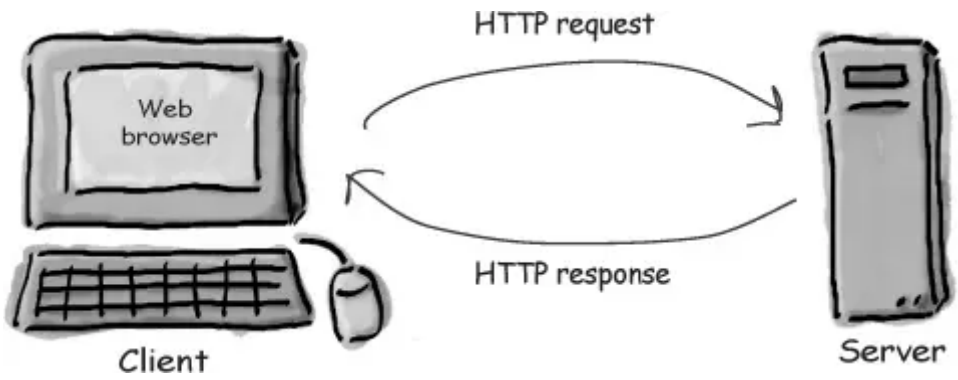
2017-05-25 前端大全

（点击上方公众号，可快速关注）

> 作者：阮一峰 （@ruanyf）
> http://www.ruanyifeng.com/blog/2016/08/http.html
> 如有好文章投稿，请点击 → 这里了解详情

HTTP 协议是互联网的基础协议，也是网页开发的必备知识，最新版本 HTTP/2 更是让它成为技术热点。

本文介绍 HTTP 协议的历史演变和设计思路。



一、HTTP/0.9

HTTP 是基于 TCP/IP 协议的应用层协议。它不涉及数据包（packet）传输，主要规定了客户端和服务器之间的通信格式，默认使用80端口。

最早版本是1991年发布的0.9版。该版本极其简单，只有一个命令GET。

```
GET /index.html
```

上面命令表示，TCP 连接（connection）建立后，客户端向服务器请求（request）网页 index.html。

协议规定，服务器只能回应HTML格式的字符串，不能回应别的格式。

```
<html>
 <body>Hello World</body>
</html>
```

服务器发送完毕，就关闭TCP连接。

二、HTTP/1.0

2.1 简介

1996年5月，HTTP/1.0 版本发布，内容大大增加。

首先，任何格式的内容都可以发送。这使得互联网不仅可以传输文字，还能传输图像、视频、二进制文件。这为互联网的大发展奠定了基础。

其次，除了GET命令，还引入了POST命令和HEAD命令，丰富了浏览器与服务器的互动手段。

再次，HTTP请求和回应的格式也变了。除了数据部分，每次通信都必须包括头信息（HTTP header），用来描述一些元数据。

其他的新增功能还包括状态码（status code）、多字符集支持、多部分发送（multi-part type）、权限（authorization）、缓存（cache）、内容编码（content encoding）等。

2.2 请求格式

下面是一个1.0版的HTTP请求的例子。

```
GET / HTTP/1.0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5)
Accept: */*
```

可以看到，这个格式与0.9版有很大变化。

第一行是请求命令，必须在尾部添加协议版本（HTTP/1.0）。后面就是多行头信息，描述客户端的情况。

2.3 回应格式

服务器的回应如下。

```
HTTP/1.0 200 OK
Content-Type: text/plain
Content-Length: 137582
Expires: Thu, 05 Dec 1997 16:00:00 GMT
```

```
Last-Modified: Wed, 5 August 1996 15:55:28 GMT
Server: Apache 0.84

<html>
  <body>Hello World</body>
</html>
```

回应的格式是”头信息 + 一个空行（\r\n） + 数据”。其中，第一行是”协议版本 + 状态码（status code） + 状态描述”。

2.4 Content-Type 字段

关于字符的编码，1.0版规定，头信息必须是 ASCII 码，后面的数据可以是任何格式。因此，服务器回应的时候，必须告诉客户端，数据是什么格式，这就是Content-Type字段的作用。

下面是一些常见的Content-Type字段的值。

- text/plain
- text/html
- text/css
- image/jpeg
- image/png
- image/svg+xml
- audio/mp4
- video/mp4
- application/javascript
- application/pdf
- application/zip
- application/atom+xml

这些数据类型总称为MIME type，每个值包括一级类型和二级类型，之间用斜杠分隔。

除了预定义的类型，厂商也可以自定义类型。

```
application/vnd.debian.binary-package
```

上面的类型表明，发送的是Debian系统的二进制数据包。

MIME type还可以在尾部使用分号，添加参数。

```
Content-Type: text/html; charset=utf-8
```

上面的类型表明，发送的是网页，而且编码是UTF-8。

客户端请求的时候，可以使用Accept字段声明自己可以接受哪些数据格式。

```
Accept: */*
```

上面代码中，客户端声明自己可以接受任何格式的数据。

MIME type不仅用在HTTP协议，还可以用在其他地方，比如HTML网页。

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<!-- 等同于 -->
<meta charset="utf-8" />
```

2.5 Content-Encoding 字段

由于发送的数据可以是任何格式，因此可以把数据压缩后再发送。Content-Encoding字段说明数据的压缩方法。

```
Content-Encoding: gzip
Content-Encoding: compress
Content-Encoding: deflate
```

客户端在请求时，用Accept-Encoding字段说明自己可以接受哪些压缩方法。

```
Accept-Encoding: gzip, deflate
```

2.6 缺点

HTTP/1.0 版的主要缺点是，每个TCP连接只能发送一个请求。发送数据完毕，连接就关闭，如果还要请求其他资源，就必须再新建一个连接。

TCP连接的新建成本很高，因为需要客户端和服务器三次握手，并且开始时发送速率较慢（slow start）。所以，HTTP 1.0版本的性能比较差。随着网页加载的外部资源越来越多，这个问题就愈发突出了。

为了解决这个问题，有些浏览器在请求时，用了一个非标准的Connection字段。

```
Connection: keep-alive
```

这个字段要求服务器不要关闭TCP连接，以便其他请求复用。服务器同样回应这个字段。

```
Connection: keep-alive
```

一个可以复用的TCP连接就建立了，直到客户端或服务器主动关闭连接。但是，这不是标准字段，不同实现的行为可能不一致，因此不是根本的解决办法。

三、HTTP/1.1

1997年1月，HTTP/1.1 版本发布，只比 1.0 版本晚了半年。它进一步完善了 HTTP 协议，一直用到了20年后的今天，直到现在还是最流行的版本。

3.1 持久连接

1.1 版的最大变化，就是引入了持久连接（persistent connection），即TCP连接默认不关闭，可以被多个请求复用，不用声明Connection: keep-alive。

客户端和服务器发现对方一段时间没有活动，就可以主动关闭连接。不过，规范的做法是，客户端在最后一个请求时，发送Connection: close，明确要求服务器关闭TCP连接。

```
Connection: close
```

目前，对于同一个域名，大多数浏览器允许同时建立6个持久连接。

3.2 管道机制

1.1 版还引入了管道机制（pipelining），即在同一个TCP连接里面，客户端可以同时发送多个请求。这样就进一步改进了HTTP协议的效率。

举例来说，客户端需要请求两个资源。以前的做法是，在同一个TCP连接里面，先发送A请求，然后等待服务器做出回应，收到后再发出B请求。管道机制则是允许浏览器同时发出A请求和B请求，但是服务器还是按照顺序，先回应A请求，完成后再回应B请求。

3.3 Content-Length 字段

一个TCP连接现在可以传送多个回应，势必就要有一种机制，区分数据包是属于哪一个回应的。这就是Content-length字段的作用，声明本次回应的数据长度。

```
Content-Length: 3495
```

上面代码告诉浏览器，本次回应的长度是3495个字节，后面的字节就属于下一个回应了。

在1.0版中，Content-Length字段不是必需的，因为浏览器发现服务器关闭了TCP连接，就表明收到的数据包已经全了。

3.4 分块传输编码

使用Content-Length字段的前提条件是，服务器发送回应之前，必须知道回应的数据长度。

对于一些很耗时的动态操作来说，这意味着，服务器要等到所有操作完成，才能发送数据，显然这样的效率不高。更好的处理方法是，产生一块数据，就发送一块，采用”流模式”（stream）取代”缓存模式”（buffer）。

因此，1.1版规定可以不使用Content-Length字段，而使用“分块传输编码”（chunked transfer encoding）。只要请求或回应的头信息有Transfer-Encoding字段，就表明回应将由数量未定的数据块组成。

```
Transfer-Encoding: chunked
```

每个非空的数据块之前，会有一个16进制的数值，表示这个块的长度。最后是一个大小为0的块，就表示本次回应的数据发送完了。下面是一个例子。

```
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked

25
This is the data in the first chunk

1C
and this is the second one

3
con

8
sequence

0
```

3.5 其他功能

1.1版还新增了许多动词方法：PUT、PATCH、HEAD、 OPTIONS、DELETE。

另外，客户端请求的头信息新增了Host字段，用来指定服务器的域名。

```
Host: www.example.com
```

有了Host字段，就可以将请求发往同一台服务器上的不同网站，为虚拟主机的兴起打下了基础。

3.6 缺点

虽然1.1版允许复用TCP连接，但是同一个TCP连接里面，所有的数据通信是按次序进行的。服务器只有处理完一个回应，才会进行下一个回应。要是前面的回应特别慢，后面就会有许多请求排队等着。这称为“队头堵塞”（Head-of-line blocking）。

为了避免这个问题，只有两种方法：一是减少请求数，二是同时多开持久连接。这导致了很多的网页优化技巧，比如合并脚本和样式表、将图片嵌入CSS代码、域名分片（domain sharding）等等。如果HTTP协议设计得更好一些，这些额外的工作是可以避免的。

四、SPDY 协议

2009年，谷歌公开了自行研发的 SPDY 协议，主要解决 HTTP/1.1 效率不高的问题。

这个协议在Chrome浏览器上证明可行以后，就被当作 HTTP/2 的基础，主要特性都在 HTTP/2 之中得到继承。

五、HTTP/2

2015年，HTTP/2 发布。它不叫 HTTP/2.0，是因为标准委员会不打算再发布子版本了，下一个新版本将是 HTTP/3。

5.1 二进制协议

HTTP/1.1 版的头信息肯定是文本（ASCII编码），数据体可以是文本，也可以是二进制。HTTP/2 则是一个彻底的二进制协议，头信息和数据体都是二进制，并且统称为”帧”（frame）：头信息帧和数据帧。

二进制协议的一个好处是，可以定义额外的帧。HTTP/2 定义了近十种帧，为将来的高级应用打好了基础。如果使用文本实现这种功能，解析数据将会变得非常麻烦，二进制解析则方便得多。

5.2 多工

HTTP/2 复用TCP连接，在一个连接里，客户端和浏览器都可以同时发送多个请求或回应，而且不用按照顺序一一对应，这样就避免了”队头堵塞”。

举例来说，在一个TCP连接里面，服务器同时收到了A请求和B请求，于是先回应A请求，结果发现处理过程非常耗时，于是就发送A请求已经处理好的部分， 接着回应B请求，完成后，再发送A请求剩下的部分。

这样双向的、实时的通信，就叫做多工（Multiplexing）。

5.3 数据流

因为 HTTP/2 的数据包是不按顺序发送的，同一个连接里面连续的数据包，可能属于不同的回应。因此，必须要对数据包做标记，指出它属于哪个回应。

HTTP/2 将每个请求或回应的所有数据包，称为一个数据流（stream）。每个数据流都有一个独一无二的编号。数据包发送的时候，都必须标记数据流ID，用来区分它属于哪个数据流。另外还规定，客户端发出的数据流，ID一律为奇数，服务器发出的，ID为偶数。

数据流发送到一半的时候，客户端和服务器都可以发送信号（RST_STREAM帧），取消这个数据流。1.1版取消数据流的唯一方法，就是关闭TCP连接。这就是说，HTTP/2 可以取消某一次请求，同时保证TCP连接还打开着，可以被其他请求使用。

客户端还可以指定数据流的优先级。优先级越高，服务器就会越早回应。

5.4 头信息压缩

HTTP 协议不带有状态，每次请求都必须附上所有信息。所以，请求的很多字段都是重复的，比如Cookie和User Agent，一模一样的内容，每次请求都必须附带，这会浪费很多带宽，也影响速度。

HTTP/2 对这一点做了优化，引入了头信息压缩机制（header compression）。一方面，头信息使用gzip或compress压缩后再发送；另一方面，客户端和服务器同时维护一张头信息表，所有字段都会存入这个表，生成一个索引号，以后就不发送同样字段了，只发送索引号，这样就提高速度了。

5.5 服务器推送

HTTP/2 允许服务器未经请求，主动向客户端发送资源，这叫做服务器推送（server push）。

常见场景是客户端请求一个网页，这个网页里面包含很多静态资源。正常情况下，客户端必须收到网页后，解析HTML源码，发现有静态资源，再发出静态资源请求。其实，服务器可以预期到客

户端请求网页后，很可能会再请求静态资源，所以就主动把这些静态资源随着网页一起发给客户端了。

六、参考链接

- Journey to HTTP/2, by Kamran Ahmed
- HTTP, by Wikipedia
- HTTP/1.0 Specification
- HTTP/2 Specification

阅读原文