第 4 章 HTTP/1.1 的语法：追求高速化和安全性(2)

[日] 涩川喜规 · 详解HTTP：协议基础与Go语言实现

4.6　支持虚拟主机

HTTP/1.0 以使用一台 Web 服务器来处理一个域为前提，然而给每个网站准备一个服务器是不现实的。HTTP/1.1 支持使用一台 Web 服务器提供多个 Web 服务。

假设我们要访问 `http://××××.com/hello` 这个 URL。先说 `××××.com` 部分。只要询问域名服务器，就可以知道持有该域的服务器的 IP 地址。然后，通过查看 `http` 部分或者域名后面的端口号（如 `××××.com:8080`）来确定端口号。在 HTTP/1.1 之前，服务器实际接收的信息只有最后的路径 `/hello`。

从 HTTP/1.1 开始，客户端要在 `Host` 首部中记述要发送请求的服务器名称。不管是使用 curl 命令，还是什么都不设置，都要加上该首部。假设使用同一个服务器的同一个端口提供 `tokyo.××××.com` 和 `osaka.××××.com` 两个服务。服务器只要查看请求首部的 `Host` 首部，就可以判断出请求的是哪一个服务。

在使用 Web 服务器 Apache 或 nginx 的情况下，服务器可以根据主机名获取并返回服务内容。Apache 的设置示例如下所示。

复制代码

```
NameVirtualHost *:80

<VirtualHost *:80>
    ServerName tokyo.××××.com
    DocumentRoot /www/tokyo
</VirtualHost>

<VirtualHost *:80>
    ServerName osaka.××××.com
    DocumentRoot /www/osaka
</VirtualHost>
```

虽然客户端只加了一个 `Host`，但服务器据此就可以分别提供不同的内容。

4.7 Chunk

HTTP 从版本 1.1 开始支持一些新的数据形式，其中有一个叫作 **Chunk** 的数据传输方式。具体来说就是将数据分成多个部分进行发送。在使用 Chunk 的情况下，比较耗时的数据传输可以提前进行。

例如，Chunk 有减轻服务器负荷的效果。在上传视频或者发送耗时的搜索结果时，Chunk 可以帮助我们从视频开头一点一点返回数据，或者按搜索引擎的查找结果依次返回数据。客户端会将 Chunk 合并到一起处理，但服务器可以只把用于传输的块加载到内存中，让数据流入 TCP 套接字。在这种情况下，发送 1 GB 的视频文件并不会消耗 1 GB 的内存。当服务器准备好最后的数据时，此前的数据已经传输完毕，对客户端来说可以缩短读取时间。如果是 JPEG、GIF 或者 PNG 等格式，服务器还可以仅显示下载的部分，或者隔行显示，由此加快响应速度。

Chunk 还可以实现客户端的最优通信[14]。在动态生成正文的页面中，如果显示耗时较长，就可以先只将首部部分分成 Chunk 并返回。浏览器可以先解析开头的 Chunk，在读取整个 HTML 文件之前，先下载脚本或 CSS 等显示所需的文件[15]。除此之外，Chunk 还应用在服务器的请求中，在保持连接的状态下，服务器可以在向客户端发送通知时发送 Chunk，相关内容会在第 11 章中介绍。

Chunk 的结构如下所示。

复制代码

```
HTTP/1.1 200 OK
Date: Wed, 16 Aug 2016 00:50:21 GMT
Content-Type: video/webm
Transfer-Encoding: chunked

186a0
 (100 KB 的数据)
186a0
 (100 KB 的数据)
186a0
 (100 KB 的数据)
0
```

主体被分成多个数据块。首先是十六进制数的文件大小，紧随其后的是指定大小的数据。RFC 中定义，当设置了 `Transfer-Encoding: chunked` 时，不可以包含 `Content-Length` 首部。数据大小是主体中指定的大小的总和。最后发送的 `0` 表示所有的 Chunk 传输完毕。

除了下载，Chunk 还可以用于上传。上传的形式与下载一样。

要想使用 curl 命令按照 Chunk 方式上传数据，需要进行文件传输（`-T`），并添加首部。curl 将数据分成 8 KB 的块进行发送。

复制代码

```
$ curl -T http10.rst -H "Transfer-Encoding: chunked" http://localhost:18888
```

浏览器无法使用 Chunk 上传数据。虽然可以在 JavaScript 上划分范围，根据范围上传数据，但这不是标准方法，所以有必要使用服务器应用程序将它们合并成一个文件。

从 HTTP/2 开始，以 Chunk 为单位的部分下载不再使用这些标签进行。在介绍 HTTP/2 的章节中会提到，HTTP/2 的数据传输是按照帧这个数据块来进行的。首部使用 `HEADER` 帧，文件主体使用 `DATA` 帧来发送。

在本书执笔时，Google 日本首页的 HTML 就是采用这种方式传输数据的。

向尾部添加首部

只有在使用 Chunk 的情况下才可以向尾部添加首部。

```
1 Trailer: Content-Type
```

客户端会通知服务器这里添加的首部会在主体发送之后发送。只有在使用 Chunk 的情况下才可以向尾部添加首部，换句话说，必须事先让服务器知道使用的是 Chunk，为此客户端不能指定需要的首部。另外，之后也不可以发送 `Trailer` 本身。因此，以下首部无法指定。

> Transfer-Encoding
>
> Content-Length
>
> Trailer

4.8 确认主体发送

客户端并不是一次性将数据发送给服务器的，而是在确认服务器能接收数据后才发送主体内容。

首先，客户端会发送下面的首部和主体之外的所有首部。即使没有文件，客户端也会将 `Content-Length` 首部一起发送给服务器。

```
1 Expect: 100-continue
```

若服务器返回如下响应，则表示服务器能接收数据，这时客户端才会发送主体内容。

```
100 Continue
```

服务器如果不支持客户端要发送的数据，则返回 `417 Expectation Failed`。

curl 命令默认发送该首部，并分两个阶段进行 `POST`。如果发送内容的文件大小超过 1025 字节，则执行该动作，否则将 `Expect` 首部设为空，具体如下所示。

```
$ curl -H "Expect:" --data-binary @bigfile.txt http://localhost:18888
```

4.9 DataURI 方案

Data URI 方案是在 RFC 2397 中定义的。在介绍 HTTP/1.0 时提到的方案用于规定连接的协议，而如果使用 Data URI 方案，URI 就不再是表示数据位置的标识符，而是数据本身。如果在浏览器的地址栏中输入下面的文本，则与返回 JSON 的 Web API 的响应一样，数据会显示在浏览器中。

```
data:application/json,{"message": "hello"}
```

`data` 后面紧跟着的就是 MIME 类型。我们可以添加可选项 `;base64`，还可以处理二进制数据。

Data URI 方案的常见用途是将图像作为文本嵌入到样式表中。在指定标签的背景图像时，我们可以通过 `url()` 来指定文件的 URL，这里可以写成 Data URI 方案。通常在读入 CSS 之后，如果发现 `url()`，浏览器会再次向网站发送请求，在下载完图像文件之前无法显示内容。在 Data URI 方案的情况下，由于 CSS 中包含所有数据，所以浏览器不会产生用于下载图像的网络等待时间。除样式表之外，`<img>` 标签的 `src` 属性中也可以设置 Data URI 方案。不过，如果采用 Base64 编码，文件大小将变为原来的 4/3 倍。

用文本表示图像的方法还有使用 JavaScript 来动态生成文件，具体内容会在介绍 JavaScript 的章节中讲解。

4.10 本章小结

本章介绍了 HTTP/1.1 中新增的元素等。从 HTTP/1.0 到 HTTP/1.1，Web 的可能性增加了。为了确保通信安全，HTTP/1.1 准备了统一的结构，还添加了 `PUT` 方法和 `DELETE` 方法，使 HTTP 可以作为数据管理协议使用。另外，随着 Web 内容的多样化，为了提高传输速度，协议本身也嵌入了各种结构。

笔者在本章用了大量笔墨介绍技术难度较高的 TLS。TLS 由最初实现的 SSL 不断发展而来。SSL 不断更新的原因是，随着计算能力的提高，密钥在短时间内被破解的可能性增大等。随着量子计算机的实用化，使密钥长度增加，让算法本身变得更加复杂的趋势不会发生改变。除此之外，笔者还介绍了虽然不进行 HTTP 通信，但看起来像进行了通信一样的 Data URI 方案。

从 1997 年 HTTP/1.1 出现到 2015 年 HTTP/2 出现，在这近 20 年间，新的技术不断涌现，其中就出现了 HTML5，还出现了许多应用程序，Web 界面也发生了很大变化。对于这些变化，HTTP/1.1 凭借其较强的扩展性和良好的灵活性进行了成功的应对。

下一章将介绍作为文件传输协议进行了较大改善的浏览器功能、支持动态 Web 的浏览器结构 XMLHttpRequest，以及 HTTP/1.1 的方法的语义的应用示例。与安全和 RESTful API 有关的内容，将会在其他章节中进行介绍。

AI智能总结

HTTP/1.1的发展在追求高速化和安全性方面取得了重要进展。其中，支持虚拟主机和Chunk数据传输方式为Web服务器提供了更高效的多Web服务支持和数据传输效率。此外，引入的向尾部添加首部和确认主体发送的特性增强了通信的灵活性和可靠性。另外，文章还介绍了Data URI方案，将数据本身作为URI，提高网页加载速度。总的来说，HTTP/1.1的这些新增元素和特性丰富了Web的可能性，提高了传输速度，同时也为数据管理和通信安全提供了更多选择。文章还提到了TLS的发展和HTTP/2的出现，以及HTML5和应用程序的涌现对

HTTP/1.1的影响。下一章将介绍浏览器功能的改善、支持动态Web的浏览器结构XMLHttpRequest，以及HTTP/1.1的方法的语义的应用示例。

[14]： 引自「chunked encoding を使った高速化の考察」（使用了分块编码的高速化研究）。

[15]： 不过，这种方法不写在 HTML 文件中，而是写在 首部中。在这种情况下，不使用分块编码也可以实现同样的操作。

精选留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。