第 3 章 HTTP 详解

超文本传输协议（HyperText Transfer Protocol，HTTP）是互联网上应用最为广泛的一种网络协议，所有的 WWW 服务器都基于该协议。HTTP 设计的最初目的是提供一种发布和接收 Web 页面的方法。

本章向读者介绍 TCP、HTTP、HTTP 资源定位、HTTP 请求及响应头详细信息、HTTP 状态码及 MIME 类型详解等。

3.1 TCP 与 HTTP

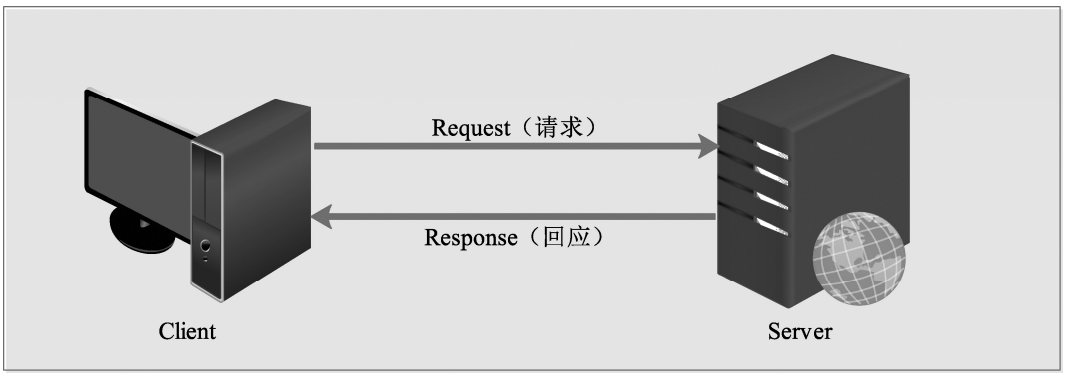
1960 年，美国人 Ted Nelson 构思了一种通过计算机处理文本信息的方法，并称为超文本（Hypertext），为 HTTP 超文本传输协议标准架构的发展奠定了根基。Ted Nelson 组织协调万维网协会（World Wide Web Consortium）和互联网工程工作小组（Internet Engineering Task Force）共同合作研究，最终发布了一系列的 RFC，其中著名的 RFC 2616 定义了 HTTP 1.1。

很多读者对 TCP 与 HTTP 存在疑问，二者有什么区别呢？从应用领域来说，TCP 主要用于数据传输控制，而 HTTP 主要用于应用层面的数据交互，本质上二者没有可比性。

HTTP 属于应用层协议，是建立在 TCP 基础之上的。HTTP 以客户端请求和服务器端应答为标准，浏览器通常称为客户端，而 Web 服务器称为服务器端。客户端打开任意一个端口向服务器端的指定端口（默认为 80）发起 HTTP 请求，首先会发起 TCP 三次握手。TCP 三次握手的目的是建立可靠的数据连接通道。TCP 三次握手通道建立完毕，即可进行 HTTP 数据交互，如图 3-1 所示。

HTTP
SSL/TLS
TCP
IP
数据链路层

（a）HTTP 与 TCP 关系结构图



（b）HTTP 客户端与服务器

图 3-1　TCP 三次握手通道建立后进行 HTTP 数据交互

当客户端请求的数据接收完毕后，HTTP 服务器端会断开 TCP 连接，整个 HTTP 连接过程非常短。HTTP 连接也称为无状态连接，无状态连接是指客户端每次向服务器发起 HTTP 请求时，每次请求都会建立一个新的 HTTP 连接，而不是在一个 HTTP 请求基础上进行所有数据的交互。

3.2　资源定位标识符

发起 HTTP 请求的内容资源由统一资源标示符（Uniform Resource Identifiers，URI）标识，关于资源定位及标识有三种：URI、URN、URL。这三种资源定位详解如下。

（1）URI（Uniform Resource Identifier，统一资源标识符），用来唯一标识一个资源。

（2）URN（Uniform Resource Name，统一资源命名），通过名字来标识或识别资源。

（3）URL（Uniform Resource Locator，统一资源定位器），是一种具体的 URI。URL 可以用来

标识一个资源，且可以访问或者获取该资源。

如图 3-2 所示，可以直观地区分 URI、URN、URL 的区别。

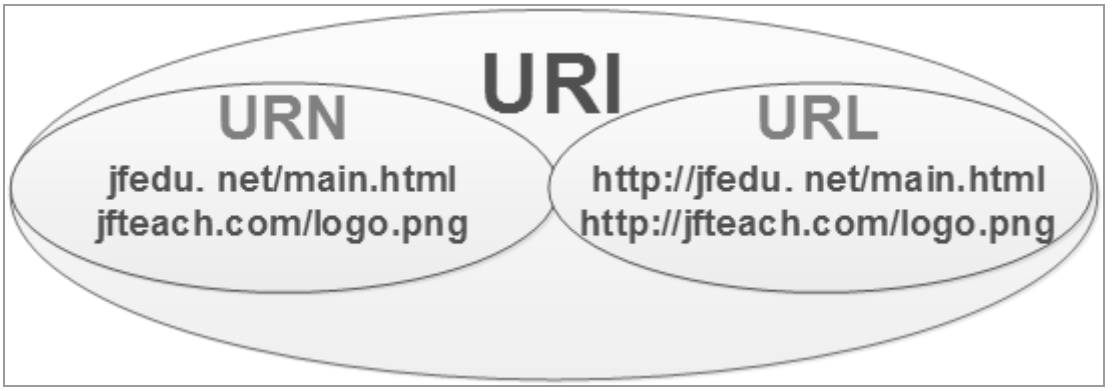


图 3-2　URI、URN、URL 的关联与区别

在这三种资源标识中，URL 资源标识方式使用最为广泛，完整的 URL 标识格式如下：

```
protocol://host[:port]/path/.../[?query-string][#anchor]
#protocol：基于某种协议,常见协议有 HTTP、HTTPS、FTP、RSYNC 等
#host：服务器的 IP 地址或者域名
#port：服务器的端口号,如果是 HTTP 80 端口,默认可以省略
#path：访问资源在服务器的路径
#query-string：传递给服务器的参数及字符串
#anchor-：锚定结束
```

HTTP URL 案例演示如下：

```
http://www.jfedu.net/newindex/plus/list.php?tid=2#jfedu
protocol:               HTTP;
host:                   www.jfedu.net;
path:                   /newindex/plus/list.php
Query String:           tid=2
Anchor:                 jfedu
```

3.3　HTTP 与端口通信

HTTP Web 服务器默认在本机会监听 80 端口。不仅 HTTP 会开启监听端口，其实每个软件程序在 Linux 系统中运行，都会以进程的方式启动，程序都会启动并监听本地接口的端口。为什么会引入端口这个概念呢?

端口是 TCP/IP 中应用层进程与传输层协议实体间的通信接口，是操作系统可分配的一种资源，应用程序通过系统调用与某个端口绑定后，传输层传给该端口的数据会被该进程接收，相应进程发给传输层的数据都通过该端口输出。

在网络通信过程中，需要唯一识别通信两端设备的端点，就是使用端口识别运行于某主机中的应用程序。如果没有引入端口，则只能通过 PID 进程号进行识别，而 PID 进程号是系统动态分配的，不同的系统会使用不同的进程标识符，应用程序在运行之前没有明确的进程号，如果需要运行后再广播进程号则很难保证通信的顺利进行。

而引入端口后，就可以利用端口号识别应用程序，同时通过固定端口号识别和使用某些公共服务，例如 HTTP 默认使用 80 端口，而 FTP 使用 21 或 20 端口，MySQL 则使用 3306 端口。

使用端口还有一个原因是随着计算机网络技术的发展，物理机器上的硬件接口已不能满足网络通信的要求，而 TCP/IP 模型作为网络通信的标准就解决了这个通信难题。

TCP/IP 中引入了一种被称为套接字（Socket）的应用程序接口。基于 Socket 接口技术，一台计算机就可以与任何一台具有 Socket 接口的计算机进行通信，而监听的端口在服务器端也被称为 Socket 接口。

3.4 HTTP Request 与 Response 详解

客户端浏览器向 Web 服务器发起 Request，Web 服务器接到 Request 后进行处理，会生成相应的 Response 信息返给浏览器，客户端浏览器收到服务器返回的 Response 信息，会对信息进行解析处理，形成最终用户看到的浏览器展示 Web 服务器的网页内容。

客户端发起的 Request 信息分为三部分，包括请求行（Request line）、请求头部（Request header）、请求数据（Body），如图 3-3 所示。

图 3-3 HTTP Request 信息组成

在 UNIX / Linux 系统中执行 CURL –v 命令可以打印访问 Web 服务器的 Request 及 Response 详细处理流程，如图 3-4 所示。

```
curl  -v  http://192.168.111.131/index.html
```

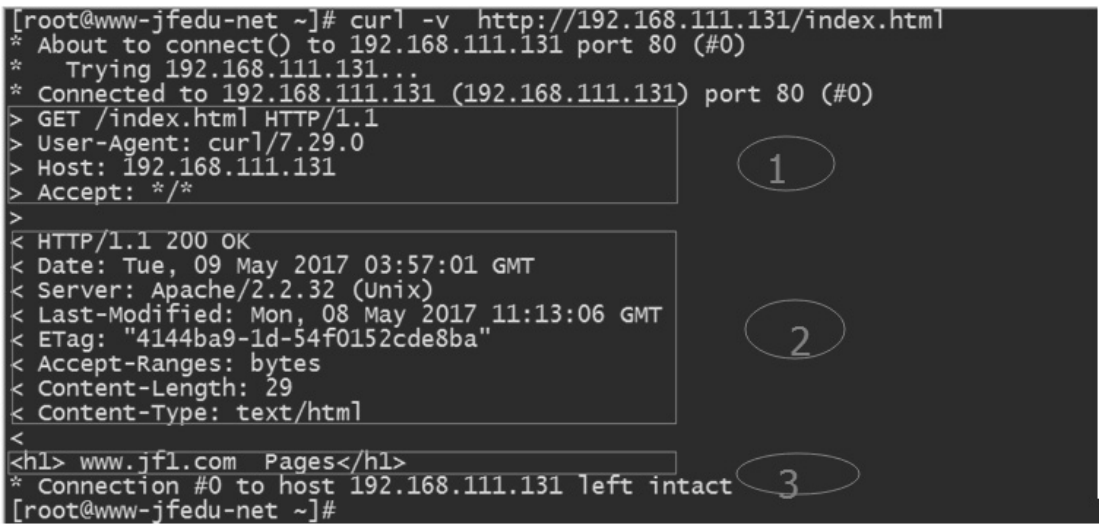


图 3-4　Request 及 Response 请求回应流程

（1）Request 信息详解如表 3-1 所示。

表 3-1　Request信息详解

GET /index.html　HTTP 1.1	请求行	Request 信息
User–Agent: curl/7.19.7 Host: 192.168.111.131 Accept: */* ……	请求头部	
>	空行	
>	请求数据	

第一部分：请求行，指定请求类型、访问的资源及使用的HTTP版本。
GET表示Request请求类型为GET；/index.html表示访问的资源；HTTP 1.1表示协议版本。
第二部分：请求头部，请求行下一行起，指定服务器要使用的附加信息。
User–Agent 表示用户使用的代理软件，常指浏览器；Host表示请求的目的主机。
第三部分：空行，请求头部后面的空行表示请求头部发送完毕。
第四部分：请求数据，又称Body，可以添加任意的数据，GET请求的Body内容默认为空。

（2）Response 信息详解如表 3-2 所示。

表 3-2　Request信息详解

HTTP 1.1 200 OK	响应状态行	Response信息
Server: Nginx 1.10.1 Date: Thu, 11 May 2021 Content-Type: text/html ……	消息报头	
>	空行	
www.jf1.com Pages	响应正文	

第一部分：响应状态行，包括HTTP版本号、状态码、状态消息。

HTTP 1.1表示HTTP版本号；200表示返回状态码；OK表示状态消息。

第二部分：消息报头，响应头部附加信息。

Date表示生成响应的日期和时间，Content-Type表示指定MIME类型的HTML（text/html），编码类型是UTF-8，记录文件资源的Last-Modified时间。

第三部分：空行，表示消息报头响应完毕。

第四部分：响应正文，服务器返回给客户端的文本信息。

（3）Request 请求方法根据请求的资源不同，有如下请求方法。

GET 方法，向特定的资源发出请求，获取服务器端数据。

POST 方法，向 Web 服务器提交数据处理请求，常指提交新数据。

PUT 方法，向 Web 服务器提交上传最新内容，常指更新数据。

DELETE 方法，请求删除 Request-URL 所标识的服务器资源。

TRACE 方法，回显服务器收到的请求，主要用于测试或诊断。

CONNECT 方法，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 方法，返回服务器针对特定资源所支持的 HTTP 请求方法。

HEAD 方法，与 GET 方法相同，只不过服务器响应时不会返回消息体。

3.5　HTTP 1.0 与 HTTP 1.1 的区别

HTTP 定义服务器端和客户端之间文件传输的沟通方式。HTTP 1.0 运行方式如图 3-5 所示。

（1）基于 HTTP 的客户/服务器模式的信息交换过程分为四个步骤，建立连接、发送请求信息、发送响应信息、关闭连接。

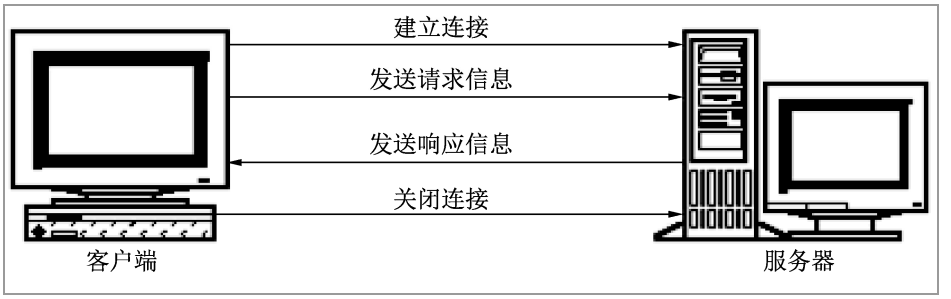


图 3-5　HTTP 1.0 客户端与服务器传输模式

（2）浏览器与 Web 服务器的连接过程是短暂的，每次连接只处理一个请求和响应。对每一个页面的访问，浏览器与 Web 服务器都要建立一次单独的连接。

（3）浏览器与 Web 服务器之间的所有通信都是完全独立分开的请求和响应。

HTTP 1.1 运行方式如图 3-6 所示。

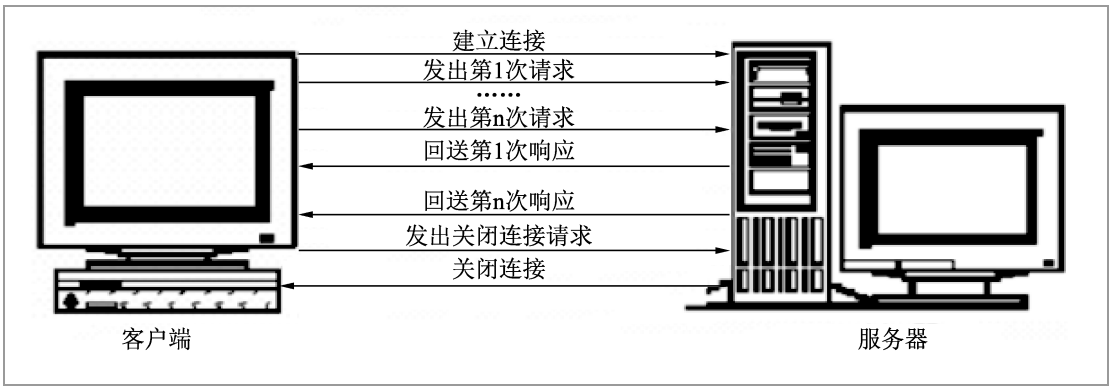


图 3-6　HTTP 1.1 客户端与服务器传输模式

（4）在一个 TCP 连接上可以传送多个 HTTP 请求和响应。

（5）多个请求和响应过程可以重叠。

（6）增加了更多的请求头和响应头，比如 Host、If-Unmodified-Since 请求头等。

3.6　HTTP 状态码详解

HTTP 状态码（HTTP Status Code）是用来表示 Web 服务器 HTTP Response 状态的 3 位数字代码，常见的状态码范围分类有以下几种。

100 ~ 199：用于指定客户端相应的某些动作。

200 ~ 299：用于表示请求成功。

300 ~ 399：已移动的文件且被包含在定位头信息中指定新的地址信息。

400 ~ 499：用于指出客户端的错误。

500 ~ 599：用于支持服务器错误。

HTTP Response 常用状态码详解如表 3-3 所示。

表 3-3　HTTP常用状态码

HTTP状态码	状态码英文含义	状态码中文含义
100	Continue	HTTP 1.1新增状态码，表示客户端继续请求HTTP服务器
101	Switching Protocols	服务器根据客户端的请求切换到HTTP的新版本协议
200	OK	HTTP请求完成，常用于GET、POST请求中
301	Moved Permanently	永久移动，请求的资源已被永久地移动到新URI
302	Found	临时移动，资源临时被移动，客户端应继续使用原有URI
304	Not Modified	文件未修改，请求的资源未修改，服务器返回此状态码时，常用于缓存
400	Bad Request	客户端请求的语法错误，服务器无法解析或者访问
401	Unauthorized	要求用户的身份认证
402	Payment Required	此状态码保留，为以后使用
403	Forbidden	服务器理解客户端的请求，但是拒绝执行该请求
404	Not Found	服务器没有该资源，请求的文件找不到
405	Method Not Allowed	客户端请求中的方法被禁止
406	Not Acceptable	服务器无法根据客户端请求的内容特性完成请求
499	Client Has Closed Connection	服务器端处理的时间过长
500	Internal Server Error	服务器内部错误，无法完成请求
502	Bad Gateway	服务器返回错误代码或代理服务器错误的网关
503	Service Unavailable	服务器无法响应客户端请求，或者后端服务器异常
504	Gateway Time-out	网关超时或者代理服务器超时
505	HTTP Version not supported	服务器不支持请求的HTTP版本，无法完成处理

3.7　HTTP MIME 类型支持

浏览器接收到 Web 服务器的 Response 信息会进行解析，在解析页面之前，浏览器必须启动

本地相应的应用程序处理获取到的文件类型。

基于多用途互联网邮件扩展类型（Multipurpose Internet Mail Extensions，MIME）可以明确某种文件在客户端用某种应用程序打开，当该扩展名文件被访问时，浏览器会自动使用指定应用程序打开，设计之初是为了在发送电子邮件时附加多媒体数据，让邮件客户程序能根据其类型进行处理。然而当它被 HTTP 支持之后，使得 HTTP 不仅可以传输普通的文本，还可以传输更多文件类型以及多媒体音、视频等。

在 HTTP 中，HTTP Response 消息的 MIME 类型被定义在 Content-Type header 中，例如，Content-Type: text/html 表示默认指定该文件为 HTML 类型，在浏览器端会以 HTML 格式处理。

在最早的 HTTP 中，并没有附加的数据类型信息，所有传送的数据都被客户程序解释为超文本标记语言 HTML 文档，为了支持多媒体数据类型，新版 HTTP 中就使用了附加在文档之前的 MIME 数据类型信息标识数据类型，如表 3-4 所示。

表 3-4　HTTP MIME类型详解

Mime-Types（MIME类型）	Dateiendung（扩展名）	Bedeutung
application/msexcel	*.xls *.xla	Microsoft Excel Dateien
application/mshelp	*.hlp *.chm	Microsoft Windows Hilfe Dateien
application/mspowerpoint	*.ppt *.ppz *.pps *.pot	Microsoft Powerpoint Dateien
application/msword	*.doc *.dot	Microsoft Word Dateien
application/octet-stream	*.exe	exe
application/pdf	*.pdf	Adobe PDF-Dateien
application/post******	*.ai *.eps *.ps	Adobe Post******-Dateien
application/rtf	*.rtf	Microsoft RTF-Dateien
application/x-httpd-php	*.php *.phtml	PHP-Dateien
application/x-java******	*.js	serverseitige Java******-Dateien
application/x-shockwave-flash	*.swf *.cab	Flash Shockwave-Dateien
application/zip	*.zip	ZIP-Archivdateien
audio/basic	*.au *.snd	Sound-Dateien
audio/mpeg	*.mp3	MPEG-Dateien
audio/x-midi	*.mid *.midi	MIDI-Dateien
audio/x-mpeg	*.mp2	MPEG-Dateien
audio/x-wav	*.wav	Wav-Dateien
image/gif	*.gif	GIF-Dateien

续表

Mime-Types（MIME类型）	Dateiendung（扩展名）	Bedeutung
image/jpeg	*.jpeg *.jpg *.jpe	JPEG-Dateien
image/x-windowdump	*.xwd	X-Windows Dump
text/css	*.css	CSS Stylesheet-Dateien
text/html	*.htm *.html *.shtml	-Dateien
text/java******	*.js	Java******-Dateien
text/plain	*.txt	reine Textdateien
video/mpeg	*.mpeg *.mpg *.mpe	MPEG-Dateien
video/vnd.rn-realvideo	*.rmvb	realplay-Dateien
video/quicktime	*.qt *.mov	Quicktime-Dateien
video/vnd.vivo	*viv *.vivo	Vivo-Dateien