章看点之一。

### （3）全世界 DNS 服务器的大接力

这时，我们的旅程进入到了 DNS 服务器帮助浏览器查询 IP 地址这一环节了。全世界共有上万台 DNS 服务器，它们相互接力才能完成 IP 地址的查询，而它们进行接力的方法也是本章看点之一。

### （4）委托协议栈发送消息

查询到 IP 地址之后，浏览器就可以将消息委托给操作系统发送给 Web 服务器了，但这个委托到底是如何完成的呢？这也是本章看点之一。“委托给操作系统”这句话看似简单，但关于委托给操作系统，其实有非常详细的规则，必须要遵守这些规则才能完成操作。由于只有编写程序的人才需要精通这些规则，所以面向一般读者的图书中几乎很少见到对这些规则的解释。不过，对这些规则有个大概的理解还是会有很多好处的，因为理解了向操作系统进行委托时的规则，我们就能够明白做出某个委托时操作系统会给我们怎样的反馈，这可以说是相当于具体地理解了网络的潜在能力。这一点对于没有编程经验的人来说也很重要。

# 1.1 生成 HTTP 请求消息

## 1.1.1 探索之旅从输入网址开始

我们的探索之旅从在浏览器中输入网址开始[①]，在介绍浏览器的工作方式之前，让我们先来介绍一下网址。网址，准确来说应该叫 URL[②]，如果我说它就是以 http:// 开头的那一串东西，恐怕大家一下子就明白了，但实际上除了“http:”，网址还可以以其他一些文字开头，例如“ftp:”“file:”“mailto:”[③]等。

之所以有各种各样的 URL，是因为尽管我们通常是使用浏览器来访问 Web 服务器的，但实际上浏览器并不只有这一个功能，它也可以用来在 FTP[④] 服务器上下载和上传文件，同时也具备电子邮件客户端的功能。可以说，浏览器是一个具备多种客户端功能的综合性客户端软件，因此它需要一些东西来判断应该使用其中哪种功能来访问相应的数据，而各种不同的 URL 就是用来干这个的，比如访问 Web 服务器时用“http:”，而访问 FTP 服务器时用“ftp:”。

图 1.1 列举了现在互联网中常见的几种 URL，根据访问目标的不同，URL 的写法也会不同。例如在访问 Web 服务器和 FTP 服务器时，URL 中会包含服务器的域名[⑤]和要访问的文件的路径名等，而发邮件的 URL 则包

---

① 某些情况下，浏览器的工作是从点击网页中的一个链接开始，大家可以认为这种情况与将链接中所包含的网址输入到浏览器的地址栏中是一样的。——译者注

② URL：Uniform Resource Locator，统一资源定位符。

③ 如果没有正确配置电子邮件软件，则即使在地址栏中输入“mailto:”也是无法正常工作的。

④ FTP：File Transfer Protocol，文件传送协议。这是一种在上传、下载文件时使用的协议。使用 FTP 协议来传送文件的程序也被叫作 FTP。

⑤ 域名：就是像 www.glasscom.com 这样以句点（.）分隔的名称。关于域名，1.2.2 节和 1.3.2 节有详细说明。

含收件人的邮件地址。此外，根据需要，URL 中还会包含用户名、密码、服务器端口号[①]等信息。

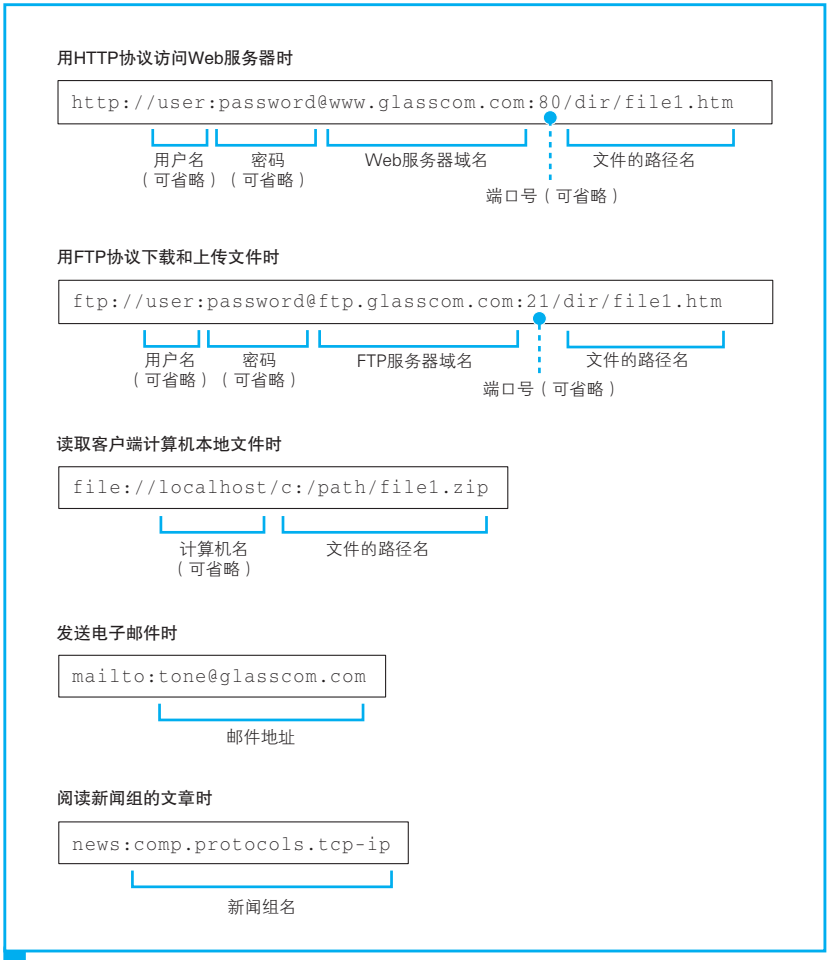


图 1.1　URL 的各种格式

① 端口号：1.4.3 节和第 6 章的 6.1.3 节有详细说明，这里请大家理解为一个用来识别要连接的服务器程序的编号。不同的服务器程序会使用不同的编号，例如 Web 是 80，邮件是 25 等。

尽管 URL 有各种不同的写法，但它们有一个共同点，那就是 URL 开头的文字，即“http:”“ftp:”“file:”“mailto:”这部分文字都表示浏览器应当使用的访问方法。比如当访问 Web 服务器时应该使用 HTTP[①] 协议，而访问 FTP 服务器时则应该使用 FTP 协议。因此，我们可以把这部分理解为访问时使用的协议[②] 类型[③]。尽管后面部分的写法各不相同，但开头部分的内容决定了后面部分的写法，因此并不会造成混乱。

### 1.1.2 浏览器先要解析 URL

浏览器要做的第一步工作就是对 URL 进行解析，从而生成发送给 Web 服务器的请求消息。刚才我们已经讲过，URL 的格式会随着协议的不同而不同，因此下面我们以访问 Web 服务器的情况为例来进行讲解。

根据 HTTP 的规格，URL 包含图 1.2（a）中的这几种元素。当对 URL 进行解析时，首先需要按照图 1.2（a）的格式将其中的各个元素拆分出来，例如图 1.2（b）中的 URL 会拆分成图 1.2（c）的样子。然后，通过拆分出来的这些元素，我们就能够明白 URL 代表的含义。例如，我们来看拆分结果图 1.2（c），其中包含 Web 服务器名称 www.lab.glasscom.com，以及文件的路径名 /dir1/file1.html，因此我们就能够明白，图 1.2（b）中的 URL 表示要访问 www.lab.glasscom.com 这个 Web 服务器上路径名为 /dir/file1.html 的文件，也就是位于 /dir/ 目录[④] 下的 file1.html 这个文件（图 1.3）。

---

① HTTP：Hypertext Transfer Protocol，超文本传送协议。

② 协议：通信操作的规则定义称为协议（protocol）。

③ 像“file:”这样的 URL 在访问时是不使用网络的，因此说 URL 的开头部分表示的是协议类型并不完全准确，也许理解为“访问方法”会更好一些。

④ 目录（directory）这个词的意思相当于 Windows 中的文件夹（folder）。

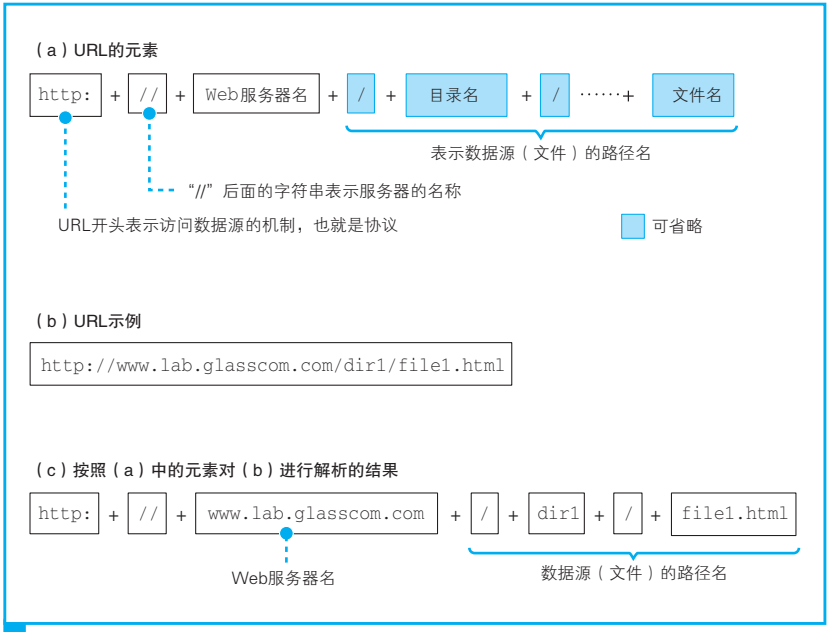


图 1.2　Web 浏览器解析 URL 的过程

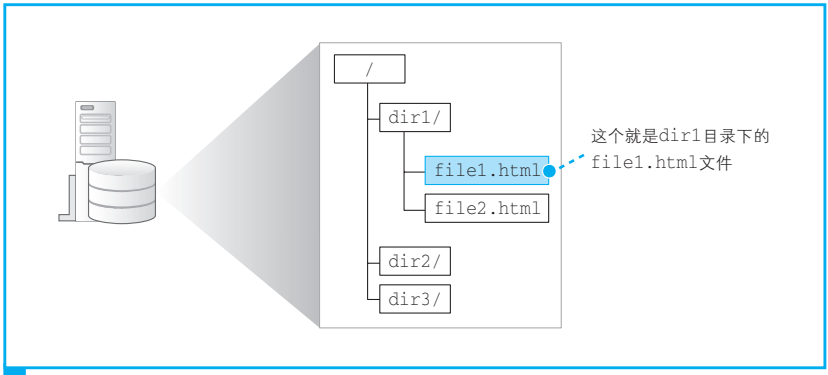


图 1.3　路径名为 /dir/file1.html 的文件

### 1.1.3 省略文件名的情况

图 1.2（b）是一个以“http:”开头的典型 URL，但有时候我们也会见到一些不太一样的 URL，例如下面这个 URL 是以“/”来结尾的。

（a）http://www.lab.glasscom.com/dir/

我们可以这样理解，以“/”结尾代表 /dir/ 后面本来应该有的文件名被省略了。根据 URL 的规则，文件名可以像前面这样省略。

不过，没有文件名，服务器怎么知道要访问哪个文件呢？其实，我们会在服务器上事先设置好文件名省略时要访问的默认文件名。这个设置根据服务器不同而不同，大多数情况下是 index.html 或者 default.htm 之类的文件名。因此，像前面这样省略文件名时，服务器就会访问 /dir/index.html 或者 /dir/default.htm。

还有一些 URL 是像下面这样只有 Web 服务器的域名的，这也是一种省略了文件名的形式。

（b）http://www.lab.glasscom.com/

这个 URL 也是以“/”结尾的，也就是说它表示访问一个名叫“/”的目录[①]。而且，由于省略了文件名，所以结果就是访问 /index.html 或者 /default.htm 这样的文件了。

那么，下面这个 URL 又是什么意思呢？

① “/”目录表示目录层级中最顶层的“根目录”。也许单独一个“/”表示根目录的写法看上去很奇怪，其实只要明白目录的规则就很容易理解了。我们写目录名时会像“dir/”一样在末尾加上一个“/”，这一点大家应该没什么疑问，那么如果目录本身没有名字的话会怎么样呢？在上面的例子中，就相当于“dir/”去掉了其中的 dir，这样一来就只剩下一个“/”了，这就是表示根目录的“/”。对于目录层级最顶层的那个目录，我们出于方便的考虑把它叫作根目录，其实它本身并没有名字，因此我们仅用“/”来表示它。

(c) http://www.lab.glasscom.com

这次连结尾的“/”都省略了。像这样连目录名都省略时，真不知道到底在请求哪个文件了，实在有些过分。不过，这种写法也是允许的。当没有路径名时，就代表访问根目录下事先设置的默认文件[①]，也就是 /index.html 或者 /default.htm 这些文件，这样就不会发生混乱了。

不过，下面这个例子就更诡异了。

(d) http://www.lab.glasscom.com/whatisthis

前面这个例子中，由于末尾没有“/”，所以 whatisthis 应该理解为文件名才对。但实际上，很多人并没有正确理解省略文件名的规则，经常会把目录末尾的“/”也给省略了。因此，或许我们不应该总是将 whatisthis 作为文件名来处理。一般来说，这种情况会按照下面的惯例进行处理：如果 Web 服务器上存在名为 whatisthis 的文件，则将 whatisthis 作为文件名来处理；如果存在名为 whatisthis 的目录，则将 whatisthis 作为目录名来处理[②]。

浏览器的第一步工作就是对 URL 进行解析。

### 1.1.4 HTTP 的基本思路

解析完 URL 之后，我们就知道应该要访问的目标在哪里了。接下来，浏览器会使用 HTTP 协议来访问 Web 服务器，不过在介绍这一环节之前，

① 最早的时候这个文件被叫作“主页”(home page)，意思就是当省略文件名时访问的那个默认的页面。随着 Web 的普及，这个词的意义似乎并没有被正确理解，现在不光是默认页面，似乎随便什么网页都可以被叫作主页了。

② 我们无法创建两个名字相同的文件和目录，因此不可能既有一个名为 whatisthis 的文件，同时又有一个名为 whatisthis 的目录。只要查询一下磁盘中的文件和目录，就可以知道 whatisthis 究竟是一个文件还是一个目录了，并不会产生歧义。

我们先来讲一讲 HTTP 协议到底是怎么回事。

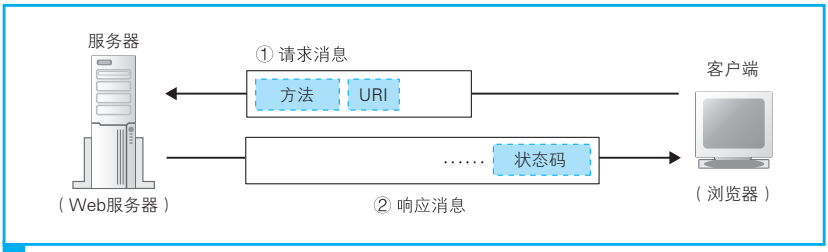


图 1.4 HTTP 的基本思路

HTTP 协议定义了客户端和服务器之间交互的消息内容和步骤，其基本思路非常简单。首先，客户端会向服务器发送请求消息（图 1.4）。请求消息中包含的内容是“对什么”和“进行怎样的操作”两个部分。其中相当于“对什么”的部分称为 URI[①]。一般来说，URI 的内容是一个存放网页数据的文件名或者是一个 CGI 程序[②]的文件名，例如“/dir1/file1.html”“/dir1/program1.cgi”等[③]。不过，URI 不仅限于此，也可以直接使用“http:”开头的 URL[④]来作为 URI。换句话说就是，这里可以写各种访问目标，而这些访问目标统称为 URI。

相当于接下来“进行怎样的操作”的部分称为方法[⑤]。方法表示需要让 Web 服务器完成怎样的工作，其中典型的例子包括读取 URI 表示的数据、将客户端输入的数据发送给 URI 表示的程序等。表 1.1 列举了主要的方法，通过这张表大家应该能够理解通过方法可以执行怎样的操作。

① URI：Uniform Resource Identifier，统一资源标识符。

② CGI 程序：对 Web 服务器程序调用其他程序的规则所做的定义就是 CGI，而按照 CGI 规范来工作的程序就称为 CGI 程序。

③ 实际上，这个文件在 Web 服务器上未必是真实存在的，因为 Web 服务器可以通过重写规则对虚拟的 URI 进行映射。——译者注

④ 5.4.3 节有详细说明。

⑤ 也叫 HTTP 谓词，或者 HTTP 动词。——译者注

表 1.1　HTTP 的主要方法

| 方法 | HTTP 版本 | | 含　义 |
|---|---|---|---|
| | 1.0 | 1.1 | |
| GET | ○ | ○ | 获取 URI 指定的信息。如果 URI 指定的是文件，则返回文件的内容；如果 URI 指定的是 CGI 程序，则返回该程序的输出数据 |
| POST | ○ | ○ | 从客户端向服务器发送数据。一般用于发送表单中填写的数据等情况下 |
| HEAD | ○ | ○ | 和 GET 基本相同。不过它只返回 HTTP 的消息头（message header），而并不返回数据的内容。用于获取文件最后更新时间等属性信息 |
| OPTIONS | | ○ | 用于通知或查询通信选项 |
| PUT | △ | ○ | 替换 URI 指定的服务器上的文件。如果 URI 指定的文件不存在，则创建该文件 |
| DELETE | △ | ○ | 删除 URI 指定的服务器上的文件 |
| TRACE | | ○ | 将服务器收到的请求行和头部（header）直接返回给客户端。用于在使用代理的环境中检查改写请求的情况 |
| CONNECT | | ○ | 使用代理传输加密消息时使用的方法 |

○：在该版本的规格中定义的项目。
△：并非正式规格，而是在规格书附录（Appendix）中定义的附加功能。
上述 1.0 版本和 1.1 版本的描述分别基于 RFC1945 和 RFC2616。

除了图 1.4 中的内容之外，HTTP 消息中还有一些用来表示附加信息的头字段。客户端向 Web 服务器发送数据时，会先发送头字段，然后再发送数据。不过，头字段属于可有可无的附加信息，因此我们留到后面再讲。

收到请求消息之后，Web 服务器会对其中的内容进行解析，通过 URI 和方法来判断“对什么”“进行怎样的操作”，并根据这些要求来完成自己的工作，然后将结果存放在响应消息中。在响应消息的开头有一个状态码，它用来表示操作的执行结果是成功还是发生了错误。当我们访问 Web 服务

器时，遇到找不到的文件就会显示出 404 Not Found 的错误信息，其实这就是状态码。状态码后面就是头字段和网页数据。响应消息会被发送回客户端，客户端收到之后，浏览器会从消息中读出所需的数据并显示在屏幕上。到这里，HTTP 的整个工作就完成了。

现在大家应该已经了解了 HTTP 的全貌，下面我们再补充一些关于 HTTP 方法的知识。表 1.1 列出的方法中，最常用的一个就是 GET 方法了。一般当我们访问 Web 服务器获取网页数据时，使用的就是 GET 方法。所谓一般的访问过程大概就是这样的：首先，在请求消息中写上 GET 方法，然后在 URI 中写上存放网页数据的文件名“/dir1/file1.html”，这就表示我们需要获取 /dir1/file1.html 文件中的数据。当 Web 服务器收到消息后，会打开 /dir1/file1.html 文件并读取出里面的数据，然后将读出的数据存放到响应消息中，并返回给客户端。最后，客户端浏览器会收到这些数据并显示在屏幕上。

还有一个经常使用的方法就是 POST。我们在表单[①]中填写数据并将其发送给 Web 服务器时就会使用这个方法。当我们在网上商城填写收货地址和姓名，或者是在网上填写问卷时，都会遇到带有输入框的网页，而这些可以输入信息的部分就是表单。使用 POST 方法时，URI 会指向 Web 服务器中运行的一个应用程序[②]的文件名，典型的例子包括“index.cgi”“index.php”等。然后，在请求消息中，除了方法和 URI 之外，还要加上传递给应用程序和脚本的数据。这里的数据也就是用户在输入框里填写的信息。当服务器收到消息后，Web 服务器会将请求消息中的数据发送给 URI 指定的应用程序。最后，Web 服务器从应用程序接收输出的结果，会将它存放到响应消息中并返回给客户端。

前面两个方法属于 HTTP 的典型用法，除此之外的其他方法在互联网上几乎见不到使用的例子。因此，只要理解了这两个方法，就能够应付大部分情况了，但如果可以，还是推荐大家看一看表 1.1 中所有方法的说明，

① 表单：网页中的文本框、复选框等能够输入数据的部分。

② 用于处理购物订单数据或者问卷数据的程序。

思考一下它们的含义，以便理解 HTTP 协议具备的所有功能。如果只有 GET 和 POST 方法，我们就只能从 Web 服务器中获取网页数据，以及将网页输入框中的信息发送给 Web 服务器，而有了 PUT 和 DELETE 方法，就能够从客户端修改或者删除 Web 服务器上的文件。有了这些功能，我们甚至可以将 Web 服务器当成文件服务器来用。当然，出于安全上的原因，或者是支持 GET 和 POST 之外的方法的客户端没有广泛普及之类的原因，一般我们并不会碰到这样的用法[①]，但大家应该能够看出，HTTP 协议其实蕴藏着很多的可能性。

### 1.1.5　生成 HTTP 请求消息

理解了 HTTP 的基本知识之后，让我们回到对浏览器本身的探索中来。

对 URL 进行解析之后，浏览器确定了 Web 服务器和文件名，接下来就是根据这些信息来生成 HTTP 请求消息了。实际上，HTTP 消息在格式上是有严格规定的，因此浏览器会按照规定的格式来生成请求消息（图 1.5）。

首先，请求消息的第一行称为请求行。这里的重点是最开头的方法，方法可以告诉 Web 服务器它应该进行怎样的操作。不过这里必须先解决一个问题，那就是方法有很多种，我们必须先判断应该选用其中的哪一种。

解决这个问题的关键在于浏览器的工作状态。这次探索之旅是从在浏览器顶部的地址栏中输入网址开始的，但浏览器并非只有在这一种场景下才会向 Web 服务器发送请求消息。比如点击网页中的超级链接[②]，或者在表单中填写信息后点击“提交”按钮，这些场景都会触发浏览器的工作，而选用哪种方法也是根据场景来确定的。

---

① 如果能够规避安全问题，例如将访问限制在公司内部网络，那么这种用法还是有效的。（实际上，PUT、DELETE 等方法现在常用于 RESTful API 的设计中，在手机 App 和后端服务器交互时就会经常用到。——译者注）

② 在 HTML 文档中写上 <a href="……"> 标签，其中 "……" 部分为 URL，这就是一个超级链接。

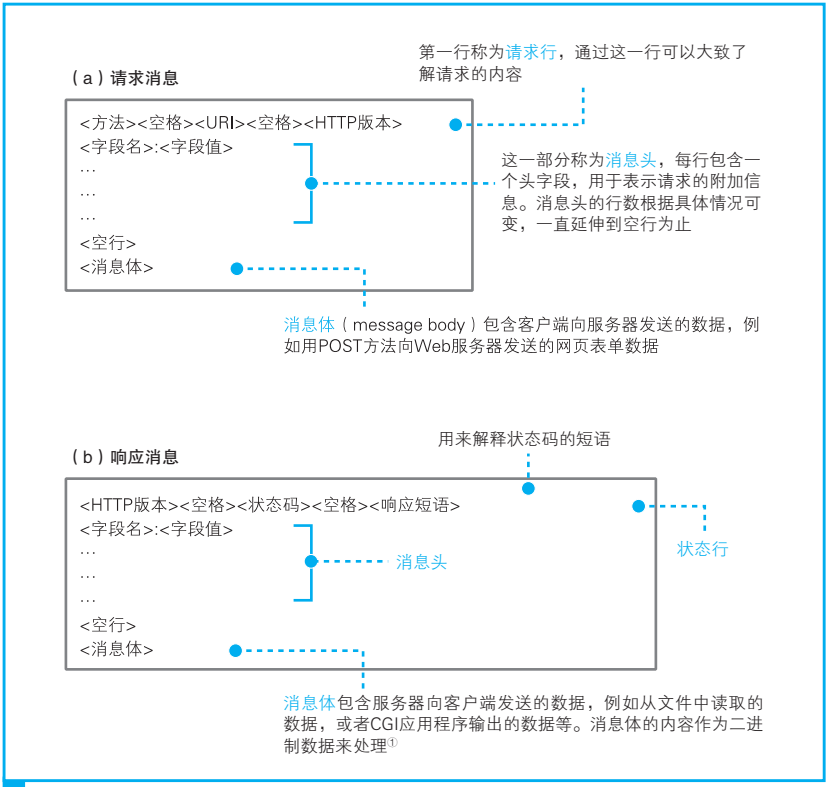


**图 1.5 HTTP 消息的格式**
浏览器和 Web 服务器根据此格式来生成消息。

我们的场景是在地址栏中输入网址并显示网页，因此这里应该使用 GET 方法。点击超级链接的场景中也是使用 GET 方法。如果是表单，在 HTML 源代码中会在表单的属性中指定使用哪种方法来发送请求，可能是 GET 也可能是 POST（图 1.6）②。

① 准确来说，消息体的格式会通过消息头中的 Content-Type 字段来定义（MIME 类型），关于 MIME 类型在本书的 6.4 节有详细介绍。——译者注

② GET 方法能够发送的数据只有几百个字节，如果表单中的数据超过这一长度，则必须使用 POST 方法来发送。

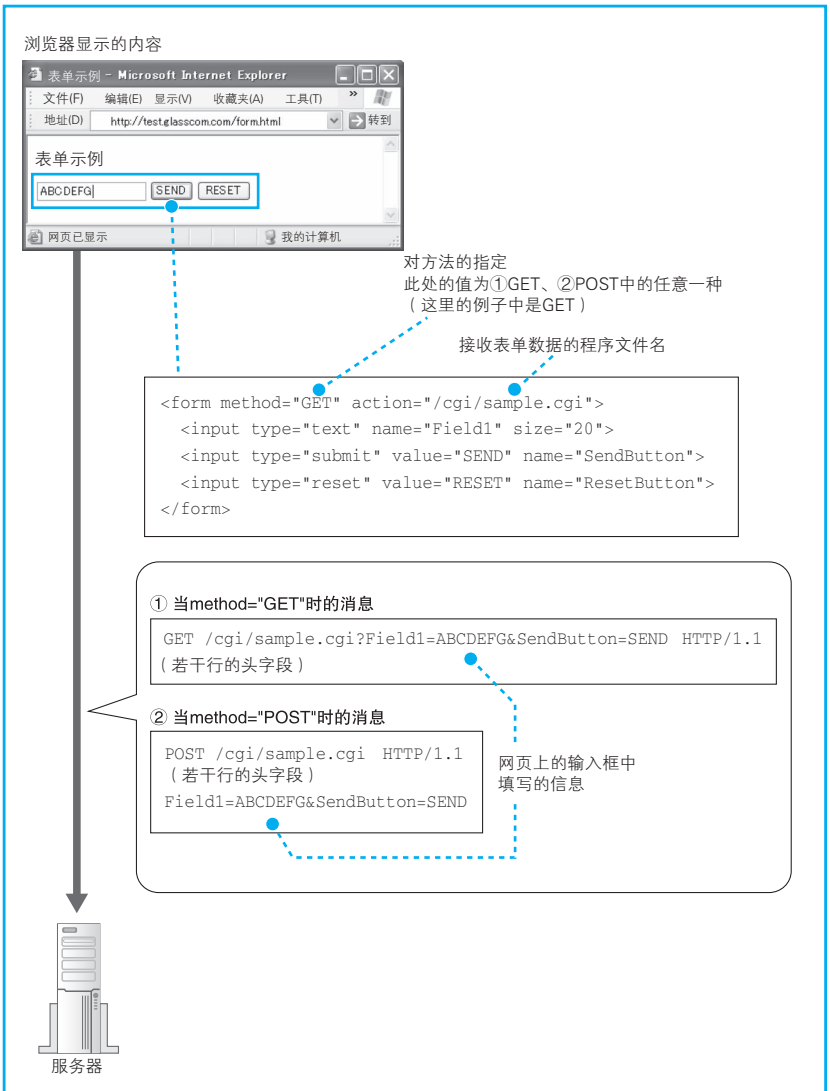


图 1.6　表单中对方法的区分

写好方法之后，加一个空格，然后写 URI。URI 部分的格式如下，一般是文件和程序的路径名。

/< 目录名 >/…/< 文件名 >

前面已经讲过，路径名一般来说已经包含在 URL 中了，因此只要从 URL 中提取出来原封不动地写上去就好了。

第一行的末尾需要写上 HTTP 的版本号，这是为了表示该消息是基于哪个版本的 HTTP 规格编写的。到此为止，第一行就结束了。

第二行开始为消息头。尽管通过第一行我们就可以大致理解请求的内容，但有些情况下还需要一些额外的详细信息，而消息头的功能就是用来存放这些信息。消息头的规格中定义了很多项目，如日期、客户端支持的数据类型、语言、压缩格式、客户端和服务器的软件名称和版本、数据有效期和最后更新时间等。这些项目表示的都是非常细节的信息，因此要想准确理解这些信息的意思，就需要对 HTTP 协议有非常深入的了解。表 1.2 中列举了主要的头字段供大家参考，但不必全部弄明白。消息头中的内容随着浏览器类型、版本号、设置等的不同而不同，大多数情况下消息头的长度为几行到十几行不等。

写完消息头之后，还需要添加一个完全没有内容的空行，然后写上需要发送的数据。这一部分称为消息体，也就是消息的主体。不过，在使用 GET 方法的情况下，仅凭方法和 URI，Web 服务器就能够判断需要进行怎样的操作，因此消息体中不需要填写任何数据。消息体结束之后，整个消息也就结束了。

表 1.2　HTTP 中主要的头字段

| 头字段类型 | HTTP 版本 | | 含　义 |
|---|---|---|---|
| | 1.0 | 1.1 | |
| 通用头：适用于请求和响应消息的头字段 | | | |
| Date | ○ | ○ | 表示请求和响应生成的日期 |
| Pragma | ○ | ○ | 表示数据是否允许缓存的通信选项 |
| Cache-Control | | ○ | 控制缓存的相关信息 |
| Connection | | ○ | 设置发送响应之后 TCP 连接是否继续保持的通信选项 |

（续）

| 头字段类型 | HTTP 版本 | | 含　义 |
|---|---|---|---|
| | 1.0 | 1.1 | |
| Transfer-Encoding | | ○ | 表示消息主体的编码格式 |
| Via | | ○ | 记录途中经过的代理和网关 |
| 请求头：用于表示请求消息的附加信息的头字段 | | | |
| Authorization | ○ | ○ | 身份认证数据 |
| From | ○ | ○ | 请求发送者的邮件地址 |
| If-Modified-Since | ○ | ○ | 如果希望仅当数据在某个日期之后有更新时才执行请求，可以在这个字段指定希望的日期。一般来说，这个功能的用途在于判断客户端缓存的数据是否已经过期，如果已经过期则获取新的数据 |
| Referer | ○ | ○ | 当通过点击超级链接进入下一个页面时，在这里会记录下上一个页面的 URI |
| User-Agent | ○ | ○ | 客户端软件的名称和版本号等相关信息 |
| Accept | △ | ○ | 客户端可支持的数据类型（Content-Type），以 MIME 类型来表示 |
| Accept-Charset | △ | ○ | 客户端可支持的字符集 |
| Accept-Encoding | △ | ○ | 客户端可支持的编码格式（Content-Encoding），一般来说表示数据的压缩格式 |
| Accept-Language | △ | ○ | 客户端可支持的语言，汉语为 zh，英语为 en |
| Host | | ○ | 接收请求的服务器 IP 地址和端口号 |
| If-Match | | ○ | 参见 Etag |
| If-None-Match | | ○ | 参见 Etag |
| If-Unmodified-Since | | ○ | 当指定日期之后数据未更新时执行请求 |
| Range | | ○ | 当需要只获取部分数据而不是全部数据时，可通过这个字段指定要获取的数据范围 |
| 响应头：用于表示响应消息的附加信息的头字段 | | | |
| Location | ○ | ○ | 表示信息的准确位置。当请求的 URI 为相对路径时，这个字段用来返回绝对路径 |
| Server | ○ | ○ | 服务器程序的名称和版本号等相关信息 |