第 2 章 Chapter 2

常见协议和标准

本章介绍 Java Web 开发中常用的协议和标准，其中包括 DNS 协议、TCP/IP 协议、HTTP 协议和 Java Web 开发中的 Servlet。

2.1 DNS 协议

DNS 协议的作用是将域名解析为 IP。我们知道网络上每个站点的位置是使用 IP 来确定的，所以要想访问一个网站首先就要知道它的 IP，不过由数字组成的 IP 记起来实在不方便，所以就设计了比较好记的域名来代替 IP，这就像我们平时看电视的时候只需要记着“中央一套”“中央二套”，而不需要去记它们是什么频率，不过实际传输还是需要按频率来传输的（对于老式用天线接收的电视来说），在我们选择了相应的频道后电视就会自动接收相应频率的数据，频道和频率的转换过程是电视机自己来处理的，但这种方式并不适合网络上的域名和 IP 的转换，首先是因为域名的数量非常多，如果让客户端的电脑自己去处理会比较慢，另外域名和 IP 的对应关系也不像电视频道那样稳定，而是经常在变化，所以就需要有专门将域名解析为 IP 的服务器，这就是“DNS 服务器”，我们把域名发过去它就可以给我们返回相应的 IP，在 Windows 中可以使用 nslookup 命令来查看 DNS 解析的结果，如使用 nslookup 命令查看淘宝的解析记录的结果如图 2-1 所示。

```
C:\Users\[redacted]>nslookup www.taobao.com
服务器:  public1.114dns.com
Address:  114.114.114.114

非权威应答:
名称:    www.gslb.taobao.com.danuoyi.tbcache.com
Addresses:  112.25.59.51
          112.25.59.41
Aliases:  www.taobao.com
```

图 2-1　使用 nslookup 命令查看淘宝 IP

从这里可以看出我现在使用的 DNS 服务器地址是 114.114.114.114，解析到 www.

taobao.com 的 IP 是 112.25.59.51 和 112.25.59.41，而且它是通过 CNAME 方式解析的，原始设置 IP 的域名是 www.gslb.taobao.com.danuoyi.tbcache.com。

世界各地有很多 DNS 服务器，ISP 会给我们提供默认的 DNS 服务器，也有一些大型公用的 DNS 服务器可以使用，比如 Google 的 8.8.8.8 和国内的 114.114.114.114。我们直接访问的 DNS 服务器叫本地 DSN 服务器，它本身也没有域名和 IP 的对应关系，在我们发出请求的时候它会从主 DNS 服务器获取然后保存到缓存中，下次再有相同的域名请求时直接从缓存中获取就可以了。

使用域名代替 IP 主要是为了方便记忆，不过域名很多时候用起来也不是那么方便，如果再加上很长的子目录和查询参数，基本就成了只有机器和专业人员才能读得懂的内容了，正因为这样导航网站才有了很大的需求。可能有人会觉得导航站主要是将键盘输入改成点击打开从而方便了操作而不是域名的问题，当然操作方式改变也是非常重要的一个因素，不过域名本身使用不方便也是非常重要的一个因素，这一点从百度指数里查看“淘宝网”的搜索量就可以看出来，同样是输入但是很多人是通过在百度搜“淘宝网”打开淘宝的而不是直接在地址栏输入 www.taobao.com 打开的。其实微信的公众号也从一定程度满足了这方面的需求。如果从这个需求出发仔细琢磨应该还有很大的发展空间。

2.2 TCP/IP 协议与 Socket

TCP/IP 协议通常放在一起来说，不过它们是两个不同的协议，所起的作用也不一样。IP 协议是用来查找地址的，对应着网际互联层，TCP 协议是用来规范传输规则的，对应着传输层。IP 只负责找到地址，具体传输的工作交给 TCP 来完成，这就像快递送货一样，货单上填写地址的规则以及怎么根据填写的内容找到客户，这就相当于 IP 协议，而送货时要先打电话，然后将货物送过去，最后客户签收时要签字等就相当于 TCP 协议。

TCP 在传输之前会进行三次沟通，一般称为“三次握手”，传完数据断开的时候要进行四次沟通，一般称为“四次挥手”。要理解这个过程首先需要理解 TCP 中的两个序号和三个标志位的含义：

- seq：sequence number 的缩写，表示所传数据的序号。TCP 传输时每一个字节都有一个序号，发送数据时会将数据的第一个序号发送给对方，接收方会按序号检查是否接收完整了，如果没接收完整就需要重新传送，这样就可以保证数据的完整性。
- ack：acknoledgement number 的缩写，表示确认号。接收端用它来给发送端反馈已经成功接收到的数据信息的，它的值为希望接收的下一个数据包起始序号，也就是 ack 值所代表的序号前面数据已经成功接收到了。
- ACK：确认位，只有 ACK=1 的时候 ack 才起作用。正常通信时 ACK 为 1，第一次发起请求时因为没有需要确认接收的数据所以 ACK 为 0。
- SYN：同步位，用于在建立连接时同步序号。刚开始建立连接时并没有历史接收的数据，所以 ack 也就没办法设置，这时按照正常的机制就无法运行了，SYN 的作用就是

来解决这个问题的，当接收端接收到 SYN=1 的报文时就会直接将 ack 设置为接收到的 seq+1 的值，注意这里的值并不是校验后设置的，而是根据 SYN 直接设置的，这样正常的机制就可以运行了，所以 SYN 叫同步位。需要注意的是，SYN 会在前两次握手时都为 1，这是因为通信的双方的 ack 都需要设置一个初始值。

- FIN：终止位，用来在数据传输完毕后释放连接。

整个传输过程如图 2-2 所示。

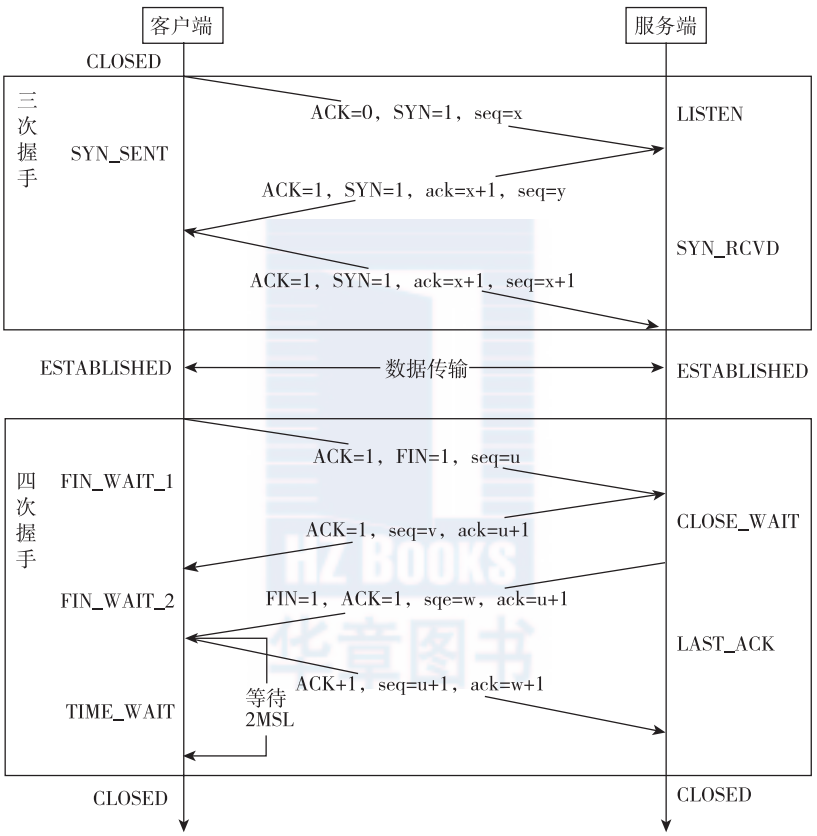

图 2-2　TCP 传输过程图

图中上部为三次握手，下部为四次挥手，这里的四次挥手中画的是客户端提出的终止连接，在实际传输过程中也有可能是服务端提出终止连接，它们的处理过程都是一样的。TCP 的传输是双全工模式，也就是说传输的双方是对等的，可以同时传输数据，所以无论连接还是关闭都需要对双方同时进行。三次握手中前两次可以保证服务端可以正确接收并返回请求，后两次可以保证客户端可以正确接收并返回请求，而且在三次握手的过程中还使用 SYN 标志初始化了双方的 ack 值。四次挥手就是双方分别发送 FIN 标志来关闭连接并让对方确认。

三次握手和四次挥手保证了连接的可靠性，不过凡事有利就有弊，这种模式也有它的缺点，首先是在传输效率上会比较低，另外三次握手的过程中客户端需要发送两次数据才可以

建立连接，这种特性可能被一些别有用心的人利用，比如，发出第一次握手（并接到第二次握手）后就不回应第三次握手了，这时服务端会以为是第二次握手的数据在传输过程中丢失了，然后重新发送第二次握手，默认情况下会一直发送五次，如果发送五次后还收不到第三次握手则会丢弃请求，如果是个别这种请求当然也没什么关系，可凡事就怕一个多字，当有大量这种请求时就麻烦了，这时服务器就会浪费大量的资源甚至可能导致无法处理正常的请求，这就是 DDOS 攻击中的 SYN Flood 攻击，对于这种攻击的一种应对方法是设置第二次请求的重发次数（tcp_synack_retries），不过重发的次数太小也可能导致正常的请求中因为网络没有收到第二次握手而连接失败的情况，具体设置为多少合适，还需要根据实际情况判断，当然如果资金充足也可以使用硬防。

用于传输层的协议除了 TCP 还有 UDP，它们的区别主要是 TCP 是有连接的，UDP 是没有连接的，也就是说 TCP 协议是在沟通好后才会传数据，而 UDP 协议是拿到地址后直就传了，这样产生的结果就是 TCP 协议传输的数据更可靠，而 UDP 传输的速度更快。TCP 就像是打电话，需要先拨通对方号码才能通信，而 UDP 就像是使用对讲机，拿起来就可以直接讲话。通常视频传输、语音传输等对完整性要求不高而对传输速度要求高并且数据量大的通信使用 UDP 比较多，而邮件、网页等一般使用 TCP 协议。

HTTP 协议的底层传输默认使用的是可靠的 TCP 协议，不过它对互联网的高速发展带来了很大的制约，Google 制定了一套基于 UDP 的 QUIC（Quick UDP Internet Connection）协议，这种协议基于 TCP 和 UDP 之间，不过现在还没有广泛使用。

TCP/IP 协议只是一套规则，并不能具体工作，就像是程序中的接口一样，而 Socket 是 TCP/IP 协议的一个具体的实现，第 3 章给大家介绍 Java 中 Socket 的具体用法。

2.3 HTTP 协议

HTTP 协议是应用层的协议，在 TCP/IP 协议接收到数据之后需要通过 HTTP 协议来解析才可以使用。就像过去的发电报一样，电报机就相当于 Socket，负责选好发送的目标并将内容发过去，但是直接发过去的数据“嘀嘀嘀”并不能直接使用，还需要解码（在发送前需要先编码再发送）后才能用，电报中的编码和解码就相当于网络传输中的 HTTP 协议。

HTTP 协议中的报文结构非常重要。HTTP 中报文分为请求报文（request message）和响应报文（response message）两种类型，这两种类型都包括三部分：首行、头部和主体。请求报文的首行是请求行，包括方法（请求类型）、URL 和 HTTP 版本三项内容，响应请求的首行是状态行，包括 HTTP 版本、状态码和简短原因三项内容，其中原因可有可无。头部保存一些键值对的属性，用冒号“：”分割。主体保存具体内容，请求报文中主要保存 POST 类型的参数，响应报文中保存页面要显示的结果。首行、头部和主体以及头部的各项内容用回车换行（\r\n）分割，另外头部和主体之间多一个空行，也就是有两个连续的回车换行。它们的结构如图 2-3 所示。

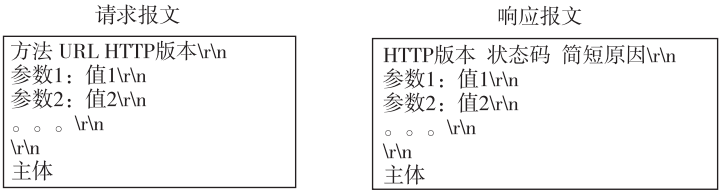

图 2-3　HTTP 报文结构

请求报文中的方法指 GET、HEAD、POST、PUT、DELETE 等类型，响应报文中的状态码就是 Response 中的 status，一共可以分为 5 类：

- 1XX：信息性状态码。
- 2XX：成功状态码，如 200 表示成功。
- 3XX：重定向状态码，如 301 表示重定向。
- 4XX：客户端错误状态码，如 404 表示没找到请求的资源。
- 5XX：服务端错误状态码，如 500 表示内部错误。

报文信息可以通过 firefox 的 firebug 插件来查看，比如，要看 www.csdn.net 网址请求的报文，可以在安装好 firefox 和 firebug 插件后按 F12 打开 firebug 的面板，然后选择“网络”下面的“HTML”，并输入网址发起请求，这时 firebug 就会记录下来，如图 2-4 所示。

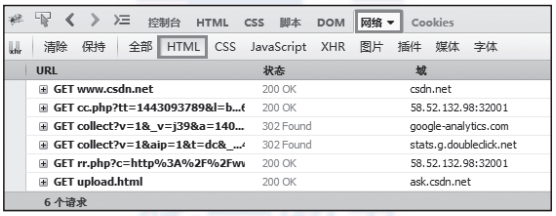

图 2-4　Firebug 记录请求

这时点击 URL 前面的加号就可以展开详细信息，如图 2-5 所示。

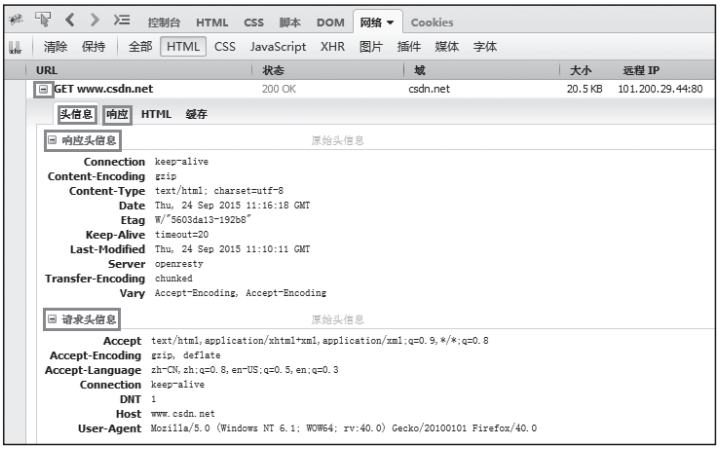

图 2-5　请求头信息

图中上边是响应的头信息，下面是请求的头信息，在“响应”选项卡中可以看到响应报文的主体。不过这时的头信息是经过格式化之后的，如果想看原始的可以点击“原始头信息”来查看，如图 2-6 所示。

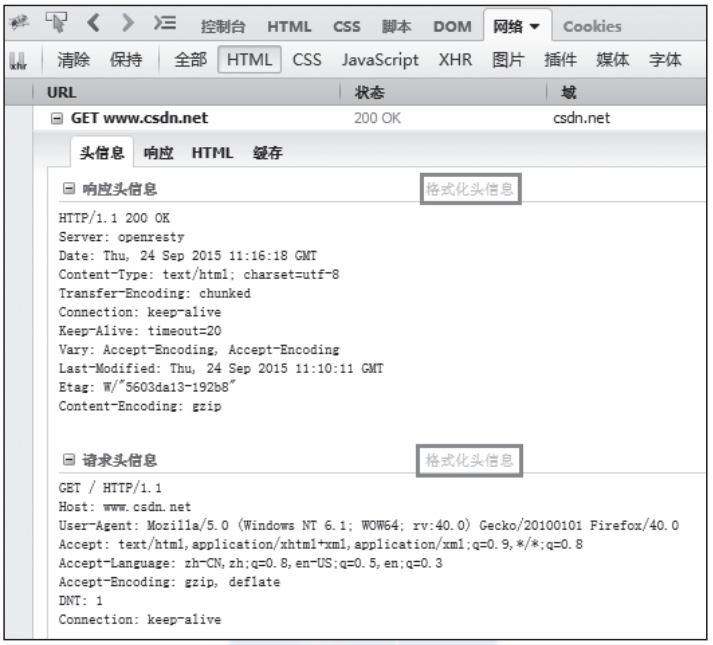

图 2-6 请求原始头信息

从这里就可以看到请求报文和响应报文的首行和头部。我们会在后面自己动手写一个实现了 HTTP 协议的简单例子。

2.4 Servlet 与 Java Web 开发

Servlet 是 J2EE 标准的一部分，是 Java Web 开发的标准。标准比协议多了强制性的意义，不过它们的作用基本是一样的，都是用来制定统一的规矩，因为 Java 是一种具体的语言，所以为了统一的实现它可以制定自己的标准。

通过前面的 TCP/IP 协议、HTTP 协议已经可以得到数据了，Servlet 的作用是对接收到的数据进行处理并生成要返回给客户端的结果，这就像电报中接收到电报并翻译成明文后还需要有人来决策并作出回复内容一样。

Servlet 制定了 Java 中处理 Web 请求的标准，我们只需要按照标准规定的去做就可以了。不过还是那句话，规范自己是不能干活的，标准一样也不能自己干活，要想使用 Servlet 需要有相应的 Servlet 容器才行，比如，我们常见的 Tomcat 就是一个 Servlet 容器，后面会给大家具体分析 Tomcat。