如何用 Java 实现 Web 服务器

徐 辉 （广西财政高等专科学校 530003）

摘要：本文介绍了 HTTP 协议的作用原理，以及根据 HTTP 协议，利用 Java 编写 Web 服务器的方法和实例。这对 Java 在 Web 中的应用，具有实践指导意义。

关键词：HTTP 协议 浏览器 Web 服务器 Java

一、HTTP 协议的作用原理

WWW 是以 Internet 作为传输媒介的一个应用系统，WWW 网上最基本的传输单位是 Web 网页。WWW 的工作基于客户机/服务器计算模型，由 Web 浏览器（客户机）和 Web 服务器（服务器）构成，两者之间采用超文本传输协议（HTTP）进行通信。HTTP 协议是基于 TCP/IP 协议之上的协议，是 Web 浏览器和 Web 服务器之间的应用层协议，是通用的、无状态的、面向对象的协议。HTTP 协议的作用原理包括四个步骤：

（1）连接：Web 浏览器与 Web 服务器建立连接，打开一个称为 socket（套接字）的虚拟文件，此文件的建立标志着连接建立成功。

（2）请求：Web 浏览器通过 socket 向 Web 服务器提交请求。HTTP 的请求一般是 GET 或 POST 命令（POST 用于 FORM 参数的传递）。GET 命令的格式为：

GET 路径/文件名 HTTP/1.0

文件名指出所访问的文件，HTTP/1.0 指出 Web 浏览器使用的 HTTP 版本。

（3）应答：Web 浏览器提交请求后，通过 HTTP 协议传送给 Web 服务器。Web 服务器接到后，进行事务处理，处理结果又通过 HTTP 传回给 Web 浏览器，从而在 Web 浏览器上显示出所请求的页面。

例：假设客户机与 www.mycompany.com:8080/mydir/index.html 建立了连接，就会发送 GET 命令：GET /mydir/index.html HTTP/1.0。主机名为 www.mycompany.com 的 Web 服务器从它的文档空间中搜索子目录 mydir 的文件 index.html。如果找到该文件，Web 服务器把该文件内容传送给相应的 Web 浏览器。

为了告知 Web 浏览器传送内容的类型，Web 服务器首先传送一些 HTTP 头信息，然后传送具体内容（即 HTTP 体信息），HTTP 头信息和 HTTP 体信息之间用一个空行分开。

常用的 HTTP 头信息有：

① HTTP 1.0 200 OK。这是 Web 服务器应答的第一行，列出服务器正在运行的 HTTP 版本号和应答代码。代码“200 OK”表示请求完成。

② MIME-Version：1.0。它指示 MIME 类型的版本。

③ content-type：类型。这个头信息非常重要，它指示 HTTP 体信息的 MIME 类型。content-type：text/html 指示传送的数据是 HTML 文档。

④ content-length：长度值。它指示 HTTP 体信息的长度（字节）。

（4）关闭连接：当应答结束后，Web 浏览器与 Web 服务器必须断开，以保证其他 Web 浏览器能够与 Web 服务器建立连接。

二、Java 实现 Web 服务器功能的程序设计

根据上述 HTTP 协议的作用原理，实现 GET 请求的 Web 服务器程序的方法如下：

（1）创建 ServerSocket 类对象，监听端口 8080。这是为了区别于 HTTP 的标准 TCP/IP 端口 80 而取的；

（2）等待、接受客户机连接到端口 8080，得到与客户机连接的 socket；

（3）创建与 socket 字相关联的输入流 instream 和输出流 outstream；

（4）从与 socket 关联的输入流 instream 中读取一行客户机提交的请求信息，请求信息的格式为：GET 路径/文件名 HTTP/1.0；

（5）从请求信息中获取请求类型。如果请求类型是 GET，则从请求信息中获取所访问的 HTML 文件名。没有 HTML 文件名时，则以 index.html 作为文件名；

(6) 如果 HTML 文件存在,则打开 HTML 文件,把 HTTP 头信息和 HTML 文件内容通过 socket 传回给 Web 浏览器,然后关闭文件。否则发送错误信息给 Web 浏览器;

(7) 关闭与相应 Web 浏览器连接的 socket 字。

下面的程序是根据上述方法编写的、可实现多线程的 Web 服务器,以保证多个客户机能同时与该 Web 服务器连接。

程序 1:WebServer.java 文件

```
//WebServer.java 用 JAVA 编写 Web 服务器
import java.io.*;
import java.net.*;

public class WebServer {
    public static void main(String args[]) {
      int i=1, PORT=8080;
      ServerSocket server=null;
      Socket client=null;
      try {
        server=new ServerSocket(PORT);
        System.out.println("Web Server is listening on port "
        +server.getLocalPort());
        for (;;) {
          client=server.accept(); //接受客户机的连接请求
          new ConnectionThread(client,i).start();
          i++;
        }
      } catch (Exception e) {System.out.println(e);}
    }
}

/* ConnnectionThread 类完成与一个 Web 浏览器的通信 */
class ConnectionThread extends Thread {
    Socket client; //连接 Web 浏览器的 socket 字
    int counter; //计数器
    public ConnectionThread(Socket cl,int c) {
      client=cl;
      counter=c;
    }
    public void run() //线程体
    {
    try {
      String destIP=client.getInetAddress().toString(); //客户机 IP 地址
      int destport=client.getPort(); //客户机端口号
      System.out.println("Connection "+counter+":connected to "+destIP+" on port "+destport+".");
      PrintStream outstream=new PrintStream(client.getOutputStream());
      DataInputStream instream=new DataInputStream(client.getInputStream());
      String inline=instream.readLine(); //读取 Web 浏览器提交的请求信息
      System.out.println("Received:"+inline);
      if (getrequest(inline)) { //如果是 GET 请求
        String filename=getfilename(inline);
        File file=new File(filename);
        if (file.exists()) { //若文件存在,则将文件送给 Web 浏览器
          System.out.println(filename+" requested.");
          outstream.println("HTTP/1.0 200 OK");
          outstream.println("MIME-version:1.0");
          outstream.println("Content-Type:text/html");
          int len=(int)file.length();
          outstream.println("Content-Length:"+len);
          outstream.println("");
          sendfile(outstream,file); //发送文件
          outstream.flush();
        } else { //文件不存在时
          String notfound="<html><head><title>Not Found</title></head>
            <body><h1>Error 404-file not found</h1></body></html>";
          outstream.println("HTTP/1.0 404 no found");
          outstream.println("Content-Type:text/html");
          outstream.println("Content-Length:"+notfound.length()+2);
          outstream.println("");
          outstream.println(notfound);
          outstream.flush();
        }
      }
      long m1=1;
      while (m1<11100000) {m1++;} //延时
      client.close();
    } catch (IOException e) {
      System.out.println("Exception:"+e);
    }
    }

/* 获取请求类型是否为"GET" */
    boolean getrequest(String s) {
      if (s.length()>0) {
        if (s.substring(0,3).equalsIgnoreCase("GET")) return true;
      }
        return false;
    }
```

```
/* 获取要访问的文件名 */
    String getfilename(String s) {
      String f = s.substring(s.indexOf(' ') + 1);
      f = f.substring(0, f.indexOf(' '));
      try {
        if (f.charAt(0) == '/')
          f = f.substring(1);
      } catch (StringIndexOutOfBoundsException e) {
        System.out.println("Exception:" + e);
      }
      if (f.equals("")) f = "index.html";
      return f;
    }

/* 把指定文件发送给 Web 浏览器 */
    void sendfile(PrintStream outs, File file) {
      try {
        DataInputStream in = new DataInputStream(new
FileInputStream(file));
        int len = (int)file.length();
        byte buf[] = new byte[len];
        in.readFully(buf);
        outs.write(buf, 0, len);
        outs.flush();
        in.close();
      } catch (Exception e) {
        System.out.println("Error retrieving file.");
        System.exit(1);
      }
    }
}
```

程序中的 ConnectionThread 线程子类用来分析一个 Web 浏览器提交的请求，并将应答信息传回给 Web 浏览器。其中 getrequest()方法用来检测客户的请求是否为“GET”；getfilename(s)方法是从客户请求信息 s 中获取要访问的 HTML 文件名；sendfile()方法把指定文件内容通过 socket 传回给 Web 浏览器。

对上述程序的 getrequest()方法和相关部分作修改，也能对 POST 请求进行处理。

三、运行实例

为了测试上述程序的正确性，将编译后的 WebServer.class、ConnectionThread.class 和下面的 index.html 文件置于网络的某台主机的同一目录中(如主机 NT40SRV 的 C:\JWEB 目录)。

程序 2：index.html 文件

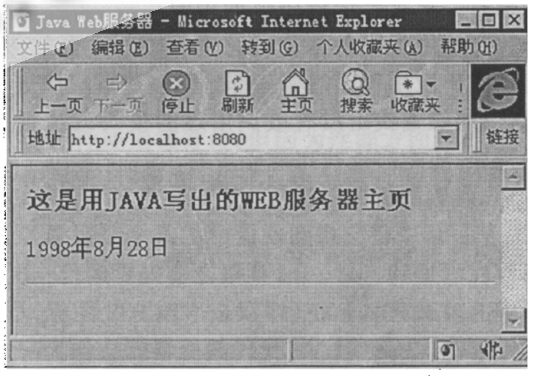
```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" content="
text/html; charset=gb-2312-80">
<TITLE>Java Web 服务器</TITLE>
</HEAD>
<BODY>
<h3>这是用 JAVA 写出的 WEB 服务器主页</h3>
    1998 年 8 月 28 日
<hr>
</BODY>
</HTML>
```

首先在该主机上用 java 命令运行 WebServer.class：

C:\jweb>java webserver

然后在客户机运行浏览器软件，在 URL 处输入 WebServer 程序所属的 URL 地址(如 http://nt40srv:8080/index.html)，就在浏览器窗口显示出指定的 HTML 文档。注意，不能缺省端口号 8080，如缺省则运行该主机的正常 WEB 服务器。

若不具备网络条件的可在安装了 Windows 95 的单机上进行测试，方法是用 localhost 或 127.0.0.1 代替 URL 地址的域名部分，即 URL 地址为 http://localhost:8080，显示效果如下图所示。



(来稿时间：1998 年 9 月)