[**Java Socket现实简单的HTTP服务**](http://jiangzhengjun.iteye.com/blog/512380)

**博客分类：**

* [Java](http://jiangzhengjun.iteye.com/category/50932)

[Socket](http://www.iteye.com/blogs/tag/Socket)[Java](http://www.iteye.com/blogs/tag/Java)[应用服务器](http://www.iteye.com/blogs/tag/%E5%BA%94%E7%94%A8%E6%9C%8D%E5%8A%A1%E5%99%A8)[.net](http://www.iteye.com/blogs/tag/.net)[浏览器](http://www.iteye.com/blogs/tag/%E6%B5%8F%E8%A7%88%E5%99%A8)

一个简单的用 Java Socket 编写的 HTTP 服务器应用, 演示了请求和应答的协议通信内容以及给客户端返回 HTML 文本和二进制数据文件(一个图片), 同时展示了 404, 200 等状态码.

首先运行这个程序,然后打开Web浏览器,键入http://localhost,则这个程序能够显示出浏览器发送了那些信息并且向浏览器返回一个网页和一副图片, 并测试同浏览器对话.  当浏览器看到 HTML 中带有图片地址时, 则会发出第二次连接来请求图片等资源.  这个例子可以帮您理解 Java 的 HTTP 服务器软件是基于 J2SE 的 Socket 等软件编写的概念, 并熟悉HTTP 协议.  相反的用 Telnet 连接到已有的服务器则可以帮忙理解浏览器的运行过程和服务器端的返回内容.    
     
当用户在Web浏览器地址栏中输入一个带有http://前缀的URL并按下Enter后,或者在Web页面中某个以http://开头的超链接上单击鼠标,HTTP事务处理的第一个阶段--建立连接阶段就开始了.HTTP的默认端口是80.  随着连接的建立,HTTP就进入了客户向服务器发送请求的阶段.客户向服务器发送的请求是一个有特定格式的ASCII消息,其语法规则为:

  < Method > < URL > < HTTP Version > <\r\n>    
  { <Header>:<Value> <\r\n>}    
  <\r\n>    
  { Entity Body }

请求消息的顶端是请求行,用于指定方法,URL和HTTP协议的版本,请求行的最后是回车换行.方法GET,POST,HEAD,PUT,DELETE等.  在请求行之后是若干个报头(Header)行.每个报头行都是由一个报头和一个取值构成的二元对,报头和取值之间以":"分隔;报头行的最后是回车换行.常见的报头有Accept(指定MIME媒体类型),Accept\_Charset(响应消息的编码方式),Accept\_Encoding(响应消息的字符集),User\_Agent(用户的浏览器信息)等.  在请求消息的报头行之后是一个回车换行,表明请求消息的报头部分结束.在这个之后是请求消息的消息实体(Entity Body).具体的例子参看httpRequest.txt.

Web服务器在收到客户请求并作出处理之后,要向客户发送应答消息.与请求消息一样,应答消息的语法规则为:

  < HTTP Version> <Status Code> [<Message>]<\r\n>    
  { <Header>:<Value> <\r\n> }     
  <\r\n>    
  { Entity Body }

应答消息的第一行为状态行,其中包括了HTTP版本号,状态码和对状态码进行简短解释的消息;状态行的最后是回车换行.状态码由3位数字组成,有5类:

  参看:HTTP应答码及其意义     
      
  1XX 保留     
  2XX 表示成功     
  3XX 表示URL已经被移走     
  4XX 表示客户错误     
  5XX 表示服务器错误

例如:415,表示不支持改媒体类型;503,表示服务器不能访问.最常见的是200,表示成功.常见的报头有:Last\_Modified(最后修改时间),Content\_Type(消息内容的MIME类型),Content\_Length(内容长度)等.

在报头行之后也是一个回车换行,用以表示应答消息的报头部分的结束,以及应答消息实体的开始.

下面是一个应答消息的例子:    
  HTTP/1.0 200 OK    
  Date: Moday,07-Apr-97 21:13:02 GMT    
  Server:NCSA/1.1    
  MIME\_Version:1.0    
  Content\_Type:text/html    
  Last\_Modified:Thu Dec 5 09:28:01 1996    
  Coentent\_Length:3107    
      
  <HTML><HEAD><TITLE></HTML>    
      
在用Java语言实现HTTP服务器时,首先启动一个java.net.ServerSocket在提供服务的端口上监听连接.向客户返回文本时,可以用 PrintWriter,但是如果返回二进制数据,则必须使用OutputStream.write(byte[])方法,返回的应答消息字符串可以使用 String.getBytes()方法转换为字节数组返回,或者使用PrintStream的print()方法写入文本,用 write(byte[])方法写入二进制数据.

源程序来自http://blog.csdn.net/myeclipse\_java，本程序是在原有基础上进行了一些调整，源程序在POST请求时会阻塞，本程序解决了此问题，另外，本程序在原有的基础上增加了文件上传与下载模拟功能：

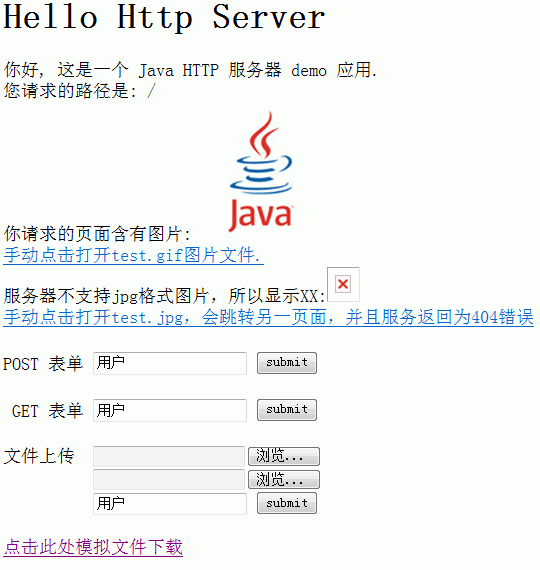
**Java代码**



1. **import** java.io.File;
2. **import** java.io.FileInputStream;
3. **import** java.io.IOException;
4. **import** java.io.InputStream;
5. **import** java.io.PrintStream;
6. **import** java.io.PrintWriter;
7. **import** java.net.ServerSocket;
8. **import** java.net.Socket;
9. **import** java.net.URLDecoder;
10. **import** java.util.ArrayList;
11. **import** java.util.StringTokenizer;
13. /\*\*
14. \*
15. \* @author 刘长炯
16. \* modi by jzj
17. \*
18. \*/
19. **public** **class** SimpleHttpServer **implements** Runnable {
21. ServerSocket serverSocket;//服务器Socket
23. **public** **static** **int** PORT = 80;//标准HTTP端口
25. **public** String encoding = "GBK";
27. **public** SimpleHttpServer() {
28. **try** {
29. serverSocket = **new** ServerSocket(PORT);
30. } **catch** (Exception e) {
31. e.printStackTrace();
32. System.exit(1);
33. }
34. **new** Thread(**this**).start();
35. System.out.println("HTTP服务器正在运行,端口:" + PORT);
36. }
38. **public** **void** run() {
39. **while** (**true**) {
40. **try** {
41. Socket client = serverSocket.accept();//客户机(这里是 IE 等浏览器)已经连接到当前服务器
42. **if** (client != **null**) {
43. System.out.println("连接到服务器的用户:" + client);
44. **try** {
45. // 第一阶段: 打开输入流
46. InputStream is = client.getInputStream();
48. System.out.println("客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>>");
49. // 读取第一行, 请求地址
50. String line = readLine(is, 0);
51. //打印请求行
52. System.out.print(line);
53. // < Method > < URL > < HTTP Version > <\r\n>  取的是URL部分
54. String resource = line.substring(line.indexOf('/'), line
55. .lastIndexOf('/') - 5);
56. //获得请求的资源的地址
57. resource = URLDecoder.decode(resource, encoding);//反编码 URL 地址
58. String method = **new** StringTokenizer(line).nextElement()
59. .toString();// 获取请求方法, GET 或者 POST
60. **int** contentLength = 0;//如果为POST方法，则会有消息体长度
62. // 读取所有浏览器发送过来的请求参数头部信息
63. **do** {
64. line = readLine(is, 0);
65. //如果有Content-Length消息头时取出
66. **if** (line.startsWith("Content-Length")) {
67. contentLength = Integer.parseInt(line.split(":")[1]
68. .trim());
69. }
70. //打印请求部信息
71. System.out.print(line);
72. //如果遇到了一个单独的回车换行，则表示请求头结束
73. } **while** (!line.equals("\r\n"));
74. //如果是POST请求，则有请求体
75. **if** ("POST".equalsIgnoreCase(method)) {
76. //注，这里只是简单的处理表单提交的参数，而对于上传文件这里是不能这样处理的，
77. //因为上传的文件时消息体不只是一行，会有多行消息体
78. System.out.print(readLine(is, contentLength));
79. System.out.println();
80. }
82. System.out.println("客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<<");
83. System.out.println("用户请求的资源是:" + resource);
84. System.out.println("请求的类型是: " + method);
85. System.out.println();
87. //如果是下载文件
88. **if** (resource.startsWith("/download")) {
89. fileDownload("test.txt", client);
90. closeSocket(client);
91. **continue**;
92. }
94. // GIF 图片就读取一个真实的图片数据并返回给客户端
95. **if** (resource.endsWith(".gif")) {
96. imgDownload("test.gif", client);
97. closeSocket(client);
98. **continue**;
99. }
101. // 请求 JPG 格式就报错 404
102. **if** (resource.endsWith(".jpg")) {
103. PrintWriter out = **new** PrintWriter(client.getOutputStream(),
104. **true**);
105. out.println("HTTP/1.0 404 Not found");//返回应答消息,并结束应答
106. out.println();// 根据 HTTP 协议, 空行将结束头信息
107. out.close();
108. closeSocket(client);
109. **continue**;
110. } **else** {
111. // 用 writer 对客户端 socket 输出一段 HTML 代码
112. PrintWriter out = **new** PrintWriter(client.getOutputStream(),
113. **true**);
114. out.println("HTTP/1.0 200 OK");//返回应答消息,并结束应答
115. out.println("Content-Type:text/html;charset=" + encoding);
116. out.println();// 根据 HTTP 协议, 空行将结束头信息
118. out.println("<h1> Hello Http Server</h1>");
119. out.println("你好, 这是一个 Java HTTP 服务器 demo 应用.<br>");
120. out.println("您请求的路径是: " + resource + "<br>");
121. out.println("你请求的页面含有图片:<img src='test.gif'><br>"
122. + "<a href='test.gif'>手动点击打开test.gif图片文件.</a>");
123. out.println("<br>服务器不支持jpg格式图片，所以显示XX:"
124. + "<img src='test.jpg'><br><a href='test.jpg'>"
125. + "手动点击打开test.jpg，会跳转另一页面，并且服务返回为404错误</a><br>");
126. out
127. .println("<form method=post action='/path?qryParm=POST URL查询参数' > POST 表单 "
128. + "<input name=username value='用户'> "
129. + "<input name=submit type=submit value=submit></form>");
130. out
131. .println("<form method=get action='/path?qryParm=GET URL查询参数' >&nbsp;GET 表单 "
132. + "<input name=username value='用户'> "
133. + "<input name=submit type=submit value=submit></form>");
135. out
136. .println("<form method=post action='/path?qryParm=POST URL查询参数'"
137. + " enctype='multipart/form-data' >"
138. + "文件上传&nbsp; <input type='file' name=file1 ><br>"
139. + "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; "
140. + "<input type='file' name=file2 ><br>"
141. + "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; "
142. + "<input name=username value='用户'> "
143. + "<input name=submit type=submit value=submit></form>");
144. out.println("<a href='/download'>点击此处模拟文件下载</a>");
146. out.close();
148. closeSocket(client);
149. }
150. } **catch** (Exception e) {
151. System.out.println("HTTP服务器错误:" + e.getLocalizedMessage());
152. }
153. }
154. //System.out.println(client+"连接到HTTP服务器");//如果加入这一句,服务器响应速度会很慢
155. } **catch** (Exception e) {
156. System.out.println("HTTP服务器错误:" + e.getLocalizedMessage());
157. }
158. }
159. }
161. /\*
162. \* 这里我们自己模拟读取一行，因为如果使用API中的BufferedReader时，它是读取到一个回车换行后
163. \* 才返回，否则如果没有读取，则一直阻塞，这就导致如果为POST请求时，表单中的元素会以消息体传送，
164. \* 这时，消息体最末按标准是没有回车换行的，如果此时还使用BufferedReader来读时，则POST提交
165. \* 时会阻塞。如果是POST提交时我们按照消息体的长度Content-Length来截取消息体，这样就不会阻塞
166. \*/
167. **private** String readLine(InputStream is, **int** contentLe) **throws** IOException {
168. ArrayList lineByteList = **new** ArrayList();
169. **byte** readByte;
170. **int** total = 0;
171. **if** (contentLe != 0) {
172. **do** {
173. readByte = (**byte**) is.read();
174. lineByteList.add(Byte.valueOf(readByte));
175. total++;
176. } **while** (total < contentLe);//消息体读还未读完
177. } **else** {
178. **do** {
179. readByte = (**byte**) is.read();
180. lineByteList.add(Byte.valueOf(readByte));
181. } **while** (readByte != 10);  //这里 10 是 ascii 码的 换行符的 值
182. }
184. **byte**[] tmpByteArr = **new** **byte**[lineByteList.size()];
185. **for** (**int** i = 0; i < lineByteList.size(); i++) {
186. tmpByteArr[i] = ((Byte) lineByteList.get(i)).byteValue();
187. }
188. lineByteList.clear();
190. String tmpStr = **new** String(tmpByteArr, encoding);
191. /\* http请求的header中有一个Referer属性，这个属性的意思就是如果当前请求是从别的页面链接过
192. \* 来的，那个属性就是那个页面的url，如果请求的url是直接从浏览器地址栏输入的就没有这个值。得
193. \* 到这个值可以实现很多有用的功能，例如防盗链，记录访问来源以及记住刚才访问的链接等。另外，浏
194. \* 览器发送这个Referer链接时好像固定用UTF-8编码的，所以在GBK下出现乱码，我们在这里纠正一下
195. \*/
196. **if** (tmpStr.startsWith("Referer")) {//如果有Referer头时，使用UTF-8编码
197. tmpStr = **new** String(tmpByteArr, "UTF-8");
198. }
199. **return** tmpStr;
200. }
202. /\*\*
203. \* 关闭客户端 socket 并打印一条调试信息.
204. \* @param socket 客户端 socket.
205. \*/
206. **void** closeSocket(Socket socket) {
207. **try** {
208. socket.close();
209. } **catch** (IOException ex) {
210. ex.printStackTrace();
211. }
212. System.out.println(socket + "离开了HTTP服务器");
213. }
215. /\*\*
216. \* 读取一个图像文件的内容并返回给浏览器端.
217. \* @param fileName 文件名
218. \* @param socket 客户端 socket.
219. \*/
220. **void** imgDownload(String fileName, Socket socket) {
222. **try** {
223. PrintStream out = **new** PrintStream(socket.getOutputStream(), **true**);
224. File fileToSend = **new** File(fileName);
225. **if** (fileToSend.exists() && !fileToSend.isDirectory()) {
226. out.println("HTTP/1.0 200 OK");//返回应答消息,并结束应答
227. out.println("Content-Type: application/octet-stream");
228. out.println("Content-Length: " + fileToSend.length());// 返回内容字节数
229. out.println();// 根据 HTTP 协议, 空行将结束头信息
231. FileInputStream fis = **new** FileInputStream(fileToSend);
232. **byte** data[] = **new** **byte**[fis.available()];
233. fis.read(data);
234. out.write(data);
235. //文件下载完后关闭socket流，但socket还没有关闭
236. out.close();
237. fis.close();
238. }
239. } **catch** (Exception e) {
240. e.printStackTrace();
241. }
242. }
244. /\*\*
245. \* 读取一个文件的内容并返回给浏览器端.
246. \* @param fileName 文件名
247. \* @param socket 客户端 socket.
248. \*/
249. **void** fileDownload(String fileName, Socket socket) {
250. **try** {
251. PrintStream out = **new** PrintStream(socket.getOutputStream(), **true**);
252. File fileToSend = **new** File(fileName);
253. **if** (fileToSend.exists() && !fileToSend.isDirectory()) {
254. out.println("HTTP/1.0 200 OK");//返回应答消息,并结束应答
255. out.println("Content-Type: application/octet-stream;charset=" + encoding);
257. /\* Content-Disposition不是标准参数，查看一下HTTP/1.1的规范文档，对于这个参数的解释大意如下：
258. \* Content-Disposition参数本来是为了在客户端另存文件时提供一个建议的文件名，但是考虑到安全的原因，
259. \* 就从规范中去掉了这个参数。但是由于很多浏览器已经能够支持这个参数，所以只是在规范文档中列出，但是要
260. \* 注意这个不是HTTP/1.1的标准参数。其值为“attachment”，那么无论这个文件是何类型，浏览器都会提示我
261. \* 们下载此文件，因为此时它认为后面的消息体是一个“附件”，不需要由浏览器来处理了。
262. \*/
263. out.println("Content-Disposition: attachment;filename=测试下载文件.txt");
264. //              out.println("Accept-Ranges: bytes");
265. out.println("Content-Length: " + fileToSend.length());// 返回内容字节数
266. out.println();// 根据 HTTP 协议, 空行将结束头信息
268. FileInputStream fis = **new** FileInputStream(fileToSend);
269. **byte**[] tmpByteArr = **new** **byte**[10];//这里为了测试看下载进度条，所以设置小点
270. **while** (fis.available() > 0) {
271. **int** readCount = fis.read(tmpByteArr);
272. out.write(tmpByteArr, 0, readCount);
273. }
275. //文件下载完后关闭socket流
276. out.close();
277. fis.close();
278. }
279. } **catch** (Exception e) {
280. e.printStackTrace();
281. }
282. }
284. **public** **static** **void** main(String[] args) {
285. PORT = 8080;
286. **new** SimpleHttpServer();
287. }
288. }

**下面是我测试的过程：**

在地址栏输入http://localhost:8080/后显示如下页面:

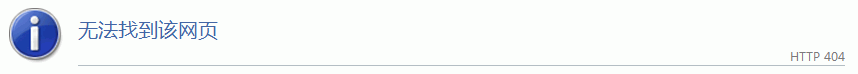


点击“手动点击打开test.gif图片文件. ”连接显示以下图片：

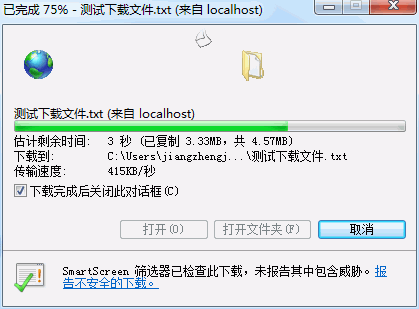


再点击“手动点击打开test.jpg，会跳转另一页面，并且服务返回为404错误”显示以下错误页面：

再点击“点击此处模拟文件下载”显示文件下载框：



   
最后是服务器运行日志，仅供参考：



|  |
| --- |
| HTTP服务器正在运行,端口:8080 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56155,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET / HTTP/1.1 Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\* Accept-Language: zh-cn User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/ 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56155,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56156,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.gif HTTP/1.1 Accept: \*/\* Referer: http://localhost:8080/ Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.gif 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56156,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56157,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.jpg HTTP/1.1 Accept: \*/\* Referer: http://localhost:8080/ Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.jpg 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56157,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56158,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.gif HTTP/1.1 Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\* Referer: http://localhost:8080/ Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.gif 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56158,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56159,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.jpg HTTP/1.1 Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\* Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.jpg 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56159,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56160,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> POST /path?qryParm=POST%20URL查询参数 HTTP/1.1 Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\* Referer: http://localhost:8080/ Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Content-Type: application/x-www-form-urlencoded Accept-Encoding: gzip, deflate Host: localhost:8080 Content-Length: 35 Connection: Keep-Alive Cache-Control: no-cache    username=%D3%C3%BB%A7&submit=submit 客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/path?qryParm=POST URL查询参数 请求的类型是: POST    Socket[addr=/0:0:0:0:0:0:0:1,port=56160,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56161,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.gif HTTP/1.1 Accept: \*/\* Referer: http://localhost:8080/path?qryParm=POST URL查询参数 Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.gif 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56161,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56162,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.jpg HTTP/1.1 Accept: \*/\* Referer: http://localhost:8080/path?qryParm=POST URL查询参数 Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.jpg 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56162,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56163,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /path?username=%D3%C3%BB%A7&submit=submit HTTP/1.1 Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\* Referer: http://localhost:8080/path?qryParm=POST URL查询参数 Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/path?username=用户&submit=submit 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56163,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56165,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.jpg HTTP/1.1 Accept: \*/\* Referer: http://localhost:8080/path?username=%D3%C3%BB%A7&submit=submit Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.jpg 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56165,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56164,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.gif HTTP/1.1 Accept: \*/\* Referer: http://localhost:8080/path?username=%D3%C3%BB%A7&submit=submit Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.gif 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56164,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56168,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> POST /path?qryParm=POST%20URL查询参数 HTTP/1.1 Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\* Referer: http://localhost:8080/path?username=%D3%C3%BB%A7&submit=submit Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Content-Type: multipart/form-data; boundary=---------------------------7d91c380444 Accept-Encoding: gzip, deflate Host: localhost:8080 Content-Length: 591 Connection: Keep-Alive Cache-Control: no-cache    -----------------------------7d91c380444 Content-Disposition: form-data; name="file1"; filename="file1.txt" Content-Type: text/plain    123 -----------------------------7d91c380444 Content-Disposition: form-data; name="file2"; filename="file2.txt" Content-Type: text/plain    这是第二个测试文件的内容: 中a ~!@#$%^&\*()\_+{}|:\" <>?`-=[]\\;',./ -----------------------------7d91c380444 Content-Disposition: form-data; name="username"    用户 -----------------------------7d91c380444 Content-Disposition: form-data; name="submit"    submit -----------------------------7d91c380444--    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/path?qryParm=POST URL查询参数 请求的类型是: POST    Socket[addr=/0:0:0:0:0:0:0:1,port=56168,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56169,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.gif HTTP/1.1 Accept: \*/\* Referer: http://localhost:8080/path?qryParm=POST URL查询参数 Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.gif 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56169,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56170,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /test.jpg HTTP/1.1 Accept: \*/\* Referer: http://localhost:8080/path?qryParm=POST URL查询参数 Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/test.jpg 请求的类型是: GET    Socket[addr=/0:0:0:0:0:0:0:1,port=56170,localport=8080]离开了HTTP服务器 连接到服务器的用户:Socket[addr=/0:0:0:0:0:0:0:1,port=56171,localport=8080] 客户端发送的请求信息: >>>>>>>>>>>>>>>>>>>>>>>>> GET /download HTTP/1.1 Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\* Referer: http://localhost:8080/path?qryParm=POST URL查询参数 Accept-Language: zh-CN User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Accept-Encoding: gzip, deflate Host: localhost:8080 Connection: Keep-Alive    客户端发送的请求信息结束 <<<<<<<<<<<<<<<<<<<<<<<<<< 用户请求的资源是:/download 请求的类型是: GET  Socket[addr=/0:0:0:0:0:0:0:1,port=56171,localport=8080]离开了HTTP服务器 |