1 public class SocketClient {  
2 OutputStream clientout = null;  
3 InputStream clienIn = null;  
4 byte[] b = new byte[1024];  
5 // String sendstr="From client 111";  
6 static String hex = "";  
7 Socket socket = null;  
8 static String hexStr = "0123456789ABCDEF";  
9 long diff1， diff2， start1， start2， end1， end2;  
10 SocketAddress socketAddress = null;  
11  
12 /\*\*  
13 \* 连接、获取流  
14 \*/  
15 public String CallClient(String serverIp， int port， String fileName，  
16 int timeout1， int timeout2， String successCode) throws IOException {  
17 try {  
18 socketAddress = new InetSocketAddress(serverIp， port);  
19 long start1 = System.currentTimeMillis();  
20 socket = new Socket();  
21 socket.connect(socketAddress， timeout1);  
22 long end1 = System.currentTimeMillis();  
23 diff1 = end1 - start1;  
24 } catch (BindException e) {  
25 //System.out.println("IP地址或端口绑定异常");  
26 socket.close();  
27 return "IP地址或端口绑定异常！";  
28 } catch (UnknownHostException e) {  
29 socket.close();  
30 return "未识别主机地址！";  
31 } catch (SocketTimeoutException e) {  
32 socket.close();  
33 return "连接超时！";  
34 } catch (ConnectException e) {  
35 socket.close();  
36 return "拒绝连接！";  
37 } catch (Exception e) {  
38 socket.close();  
39 return "连接失败！";  
40 }  
41  
42 try {  
43 System.out.println("连接建立:端口" + socket.getLocalPort());  
44 clientout = socket.getOutputStream();  
45 clienIn = socket.getInputStream();  
46 hex = file(fileName);  
47 byte[] buffer = HexStringToBinary(hex);  
48 clientout.write(buffer);  
49 // for (byte b : buffer) {  
50 // System.out.print(b);  
51 // }  
52 clientout.flush();  
53 } catch (UnknownHostException e) {  
54 e.printStackTrace();  
55 } catch (IOException e) {  
56 e.printStackTrace();  
57 }  
58  
59 /\*\*  
60 \* 返回response  
61 \*/  
62 Reader reader = new InputStreamReader(clienIn);  
63 char chars[] = new char[64];  
64 int len;  
65 StringBuffer sb = new StringBuffer();  
66 try {  
67 socket.setSoTimeout(timeout2);  
68 while ((len = reader.read(chars)) != -1) {  
69 sb.append(new String(chars， 0， len));  
70 }  
71 } catch (SocketTimeoutException e) {  
72 socket.close();  
73 return "Socket is closed";  
74 }  
75 catch (IllegalArgumentException e) {  
76 socket.close();  
77 return "timeout can't be negative";  
78 }  
79 clienIn.close();  
80 clientout.close();  
81 socket.close();  
82 String resString = sb.toString();  
83 if (resString.contains(successCode)) {  
84 System.out.println("true");  
85 return successCode;  
86 } else {  
87 System.err.println("false");  
88 return resString;  
89 }  
90 }  
91  
92 /\*\*  
93 \* 数组转换成十六进制字符串  
94 \*  
95 \* @param byte[]  
96 \* @return HexString  
97 \*/  
98 public static final String bytesToHexString(byte[] bArray) {  
99 StringBuffer sb = new StringBuffer(bArray.length);  
100 String sTemp;  
101 for (int i = 0; i < bArray.length; i++) {  
102 sTemp = Integer.toHexString(0xFF & bArray[i]);  
103 if (sTemp.length() < 2)  
104 sb.append(0);  
105 sb.append(sTemp.toUpperCase());  
106 }  
107 return sb.toString();  
108 }  
109  
110 /\*\*  
111 \* 报文处理  
112 \*  
113 \* @param hexString  
114 \* @return 将十六进制字符转换为字节数组  
115 \*/  
116 public static byte[] HexStringToBinary(String hexString) {  
117 // hexString的长度对2取整，作为bytes的长度  
118 //    System.out.println("发送报文:" + hexString);  
119 int len = hexString.length() / 2;  
120 byte[] bytes = new byte[len];  
121 byte high = 0;// 字节高四位  
122 byte low = 0; // 字节低四位  
123 for (int i = 0; i < len; i++) {  
124 // 右移四位得到高位  
125 high = (byte) ((hexStr.indexOf(hexString.charAt(2 \* i))) << 4);  
126 low = (byte) hexStr.indexOf(hexString.charAt(2 \* i + 1));  
127 bytes[i] = (byte) (high | low);// 高地位做或运算  
128 }  
129 return bytes;  
130 }  
131  
132 /\*\*  
133 \* 从文件中获取报文text  
134 \*  
135 \* @return  
136 \*/  
137 public String file(String filename) {  
138 File file = new File(filename);  
139 BufferedReader reader = null;  
140 StringBuffer sb = new StringBuffer();  
141 try {  
142 reader = new BufferedReader(new FileReader(file));  
143 String tempString = null;  
144 // 一次读入一行，直到读入null为文件结束  
145 while ((tempString = reader.readLine()) != null) {  
146 // 显示行号  
147 sb = sb.append(tempString);  
148 }  
149 reader.close();  
150 } catch (FileNotFoundException e) {  
151 e.printStackTrace();  
152 } catch (IOException e) {  
153 e.printStackTrace();  
154 }  
155 return sb.toString().replace(" "， "").replace(" "， "").replace(" "， "");  
156 }  
157 }  
158  
159  
160  
161 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
162  
163 jmeter调用部分  
164  
165 /getDefaultParameters方法：主要用于设置传入的[**参数**](javascript:;)  
166 //setupTest方法：为初始化方法，用于初始化[**性能测试**](javascript:;)时的每个线程  
167 //runTest方法：为性能测试时的线程运行体  
168 //teardownTest方法：为测试结束方法，用于结束性能测试中的每个线程  
169  
170 public class jmeter extends AbstractJavaSamplerClient {  
171  
172 private String serverIp;  
173 private int serverPort;  
174 private String file;  
175 private int timeout1;//连接超时  
176 private int timeout2;//接收超时  
177 private String successCode;  
178 @Override  
179 //runTest方法：为性能测试时的线程运行体  
180 public SampleResult runTest(JavaSamplerContext sc){  
181 serverIp=sc.getParameter("serverIp");  
182 serverPort=Integer.valueOf(sc.getParameter("serverPort"));  
183 file=sc.getParameter("file");  
184 timeout1=Integer.valueOf(sc.getParameter("timeout1"));  
185 timeout2=Integer.valueOf(sc.getParameter("timeout2"));  
186 successCode=sc.getParameter("successCode");  
187  
188 SampleResult sr=new SampleResult();  
189 sr.setSampleLabel("Socket Test");  
190 try{  
191 sr.sampleStart();//jmeter开始统计响应时间标记  
192 String receive;  
193 SocketClient c=new SocketClient();  
194 receive=c.CallClient(serverIp，serverPort，file，timeout1，timeout2，successCode);  
195 // System.out.println(receive);  
196 if (receive.equals(successCode)) {  
197 sr.setResponseMessage("返回:"+receive.toString());  
198 //System.out.println("response："+receive.toString());  
199 sr.setDataType(SampleResult.TEXT);  
200 sr.setSuccessful(true);  
201 }  
202 else {  
203 sr.setResponseMessage("报错报文："+receive.toString());  
204 sr.setSuccessful(false);  
205 }  
206 }  
207 catch(Exception e){  
208 e.printStackTrace();  
209 sr.setSuccessful(false);  
210 }  
211 finally{  
212 sr.sampleEnd();//jmeter计时结束  
213 }  
214 return sr;  
215 }  
216 @Override  
217 //getDefaultParameters方法：主要用于设置传入的参数  
218 public Arguments getDefaultParameters(){  
219 Arguments params=new Arguments();  
220 params.addArgument("serverIp"， "127.0.0.1");//设置参数，并赋予默认值1  
221 params.addArgument("serverPort"， "8009");//设置参数，并赋予默认值2  
222 params.addArgument("file"， "D:/EL04.tlt");  
223 params.addArgument("timeout1"， "10000");  
224 params.addArgument("timeout2"， "10000");  
225 params.addArgument("successCode"， "AAAAAA");  
226 return params;  
227 }  
228 @Override  
229 public void setupTest(JavaSamplerContext context){  
230 super.setupTest(context);  
231 }  
232 @Override  
233 public void teardownTest(JavaSamplerContext context){  
234 super.teardownTest(context);  
235 }  
236  
237 /\*\*\*  
238 \* 测试接口  
239 \* @param args  
240 \*/  
241 public static void main(String[] args)  
242 {  
243 Arguments params = new Arguments();  
244 params.addArgument("serverIp"， "127.0.0.1");//设置参数，并赋予默认值1  
245 params.addArgument("serverPort"， "8002");//设置参数，并赋予默认值2  
246 params.addArgument("file"， "D:/EL04.tlt");  
247 params.addArgument("timeout1"， "10000");  
248 params.addArgument("timeout2"， "10000");  
249 params.addArgument("successCode"， "AAAAAA");  
250 JavaSamplerContext arg0 = new JavaSamplerContext(params);  
251 jmeter test = new jmeter();  
252 test.setupTest(arg0);  
253 test.runTest(arg0);  
254 test.teardownTest(arg0);  
255 }  
256  
257 }