26 网络编程_TCP协议编程

JAVAEE高级

一：客户端Socket四个基本的步骤

1、 ：创建 客户Socket

> 1、根据指定服务端的 IP 地址或端口号构造 Socket 类对象
>
> 2、若服务器端响应，则建立客户端到服务器的通信线路
>
> 3、若连接失败，会出现异常。

2、连接到 Socket 的输入/出流：

> 1、使用 getInputStream()方法获得输入流，使用 getOutputStream()方法获得输出流，进行数据传输

3、一定的协议对 Socket 进行读/写操作

> 1、通过输入流读取服务器放入线路的信息（但不能读取自己放入线路的信息），通过输出流将信息写入线程

4、关闭 Socket

断开客户端到服务器的连接，释放线路

```
public class JavaTest02 {
    Socket socket = null;
    BufferedReader bufferedReader = null;
```

```
4.      PrintWriter printWriter = null;
5.      BufferedReader bufferedReader1 = null;
6.      @Test
7.      public void ClientSocket() {
8.          try {
9.              System.out.println("等待发送.............");
10.             // 设置客户端的IP及连接服务器的端口
11.             socket = new Socket("127.0.0.1", 65535);
12.             // 捕捉发送给服务器的信息
13.             bufferedReader = new BufferedReader(new InputStreamReader(S
    ystem.in));
14.             String str = bufferedReader.readLine();
15.             // 捕捉发送给服务器的信息，打包
16.             printWriter = new PrintWriter(socket.getOutputStream());
17.             // 接收服务器的反馈信息
18.             bufferedReader1 = new BufferedReader(new InputStreamReader(
    socket.getInputStream()));
19.             // 循环模式
20.             while (true) {
21.                 // 捕捉发送给服务器的信息，打包
22.                 printWriter.println(str);
23.                 // 跟催
24.                 printWriter.flush();
25.                 // 接收服务器的反馈信息
26.                 str = bufferedReader1.readLine();
27.                 System.out.println("客户收到:" + str);
28.                 // 捕捉发送给服务器的信息
29.                 str = bufferedReader.readLine();
30.             }
31.         } catch (Exception e) {
32.             // TODO Auto-generated catch block
33.             e.printStackTrace();
34.         } finally {
35.             try {
36.                 // 关闭接口
37.                 bufferedReader1.close();
38.                 printWriter.close();
39.                 bufferedReader.close();
40.                 socket.close();
41.             } catch (IOException e) {
42.                 // TODO Auto-generated catch block
43.                 e.printStackTrace();
44.             }
45.         }
46.
```

```
        }
    }
```

二：服务端ServerSocket

1、调用 ServerSocket(int port)

> 1、创建一个服务器端套接，并绑定到指定端口上
> 2、用于监听客户端的请求。

2、调用 accept()

> 1、监听连接请求，如果客户端请求连接，则接受连接，返回通信套接字对象

3、调用 该Socket类对象

> 1、getOutputStream() 和 getInputStream ()：获取输出流和输入流，开始网络数据的发送和接收

4、关闭

> 1、ServerSocket和Socket对象：客户端访问结束，关闭通信

```
public class JavaTest01 {
    ServerSocket serverSocket = null;
    Socket socket = null;
    BufferedReader bufferedReader = null;
    BufferedReader bufferedReader1 = null;
    PrintWriter printWriter = null;

    @Test
    public void MyServerSocket() {
        try {
```

```
                // 设置服务器的端口65535
                serverSocket = new ServerSocket(65535);
                System.out.println("等待请求.....");
                // 设置监听客户端
                socket = serverSocket.accept();
                // 接收客户端的信息
                String str;
                bufferedReader = new BufferedReader(new InputStreamReader(s
ocket.getInputStream()));
                // 服务器反悔给客户端信息
                bufferedReader1 = new BufferedReader(new InputStreamReader(
System.in));
                // 服务器反悔给客户端信息 打包
                printWriter = new PrintWriter(socket.getOutputStream());
                // 所有信息循环
                while (true) {
                    // 接收客户端的信息
                    str = bufferedReader.readLine();
                    System.out.println("服务器收到:" + str);
                    // 服务器反悔给客户端信息
                    str = bufferedReader1.readLine();
                    // 服务器反悔给客户端信息 打包
                    printWriter.println(str);
                    printWriter.flush();
                }
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } finally {
                try {
                    // 关闭接口
                    printWriter.close();
                    bufferedReader1.close();
                    bufferedReader.close();
                    socket.close();
                    serverSocket.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }
```