# TCP编程

## TCP传输-客户端思路

1:建立客户端的Socket服务,并明确要连接的服务器。

2:如果连接建立成功,就表明,已经建立了数据传输的通道.就可以在该通道通过IO进行数据的读取和写入.该通道称为Socket流,Socket流中既有读取流,也有写入流.

3:通过Socket对象的方法,可以获取这两个流

4:通过流的对象可以对数据进行传输

5:如果传输数据完毕,关闭资源

/\*

\* TCP协议发送数据：

\* A:创建发送端的Socket对象

\* 这一步如果成功，就说明连接已经建立成功了。

\* B:获取输出流，写数据

\* C:释放资源

\*

\* 连接被拒绝。TCP协议一定要先看服务器。

\* java.net.ConnectException: Connection refused: connect

\*/

**public** **class** ClientDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

// 创建发送端的Socket对象

// Socket(InetAddress address, int port)

// Socket(String host, int port)

// Socket s = new Socket(InetAddress.getByName("192.168.0.1"), 8888);

Socket s = **new** Socket("192.168.1.2", 12306);

// 获取输出流，写数据

// public OutputStream getOutputStream()

OutputStream os = s.getOutputStream();

os.write("hello,tcp,我来了".getBytes());

// 释放资源

s.close();

}

}

## TCP传输-服务器端思路

1:建立服务器端的socket服务，需要一个端口

2:服务端没有直接流的操作,而是通过accept方法获取客户端对象，在通过获取到的客户端对象的流和客户端进行通信

3:通过客户端的获取流对象的方法,读取数据或者写入数据

4:如果服务完成,需要关闭客户端,然后关闭服务器，但是,一般会关闭客户端,不会关闭服务器,因为服务端是一直提供服务的

/\*

\* TCP协议接收数据：

\* A:创建接收端的Socket对象

\* B:监听客户端连接。返回一个对应的Socket对象

\* C:获取输入流，读取数据显示在控制台

\* D:释放资源

\*/

**public** **class** ServerDemo {

**public** **static** **void** main(String[] args) **throws** IOException {

// 创建接收端的Socket对象

// ServerSocket(int port)

ServerSocket ss = **new** ServerSocket(12306);

// 监听客户端连接。返回一个对应的Socket对象

// public Socket accept()

Socket s = ss.accept(); // 侦听并接受到此套接字的连接。此方法在连接传入之前一直阻塞。

// 获取输入流，读取数据显示在控制台

InputStream is = s.getInputStream();

**byte**[] bys = **new** **byte**[1024];

**int** len = is.read(bys); // 阻塞式方法

String str = **new** String(bys, 0, len);

String ip = s.getInetAddress().getHostAddress();

System.*out*.println(ip + "---" + str);

// 释放资源

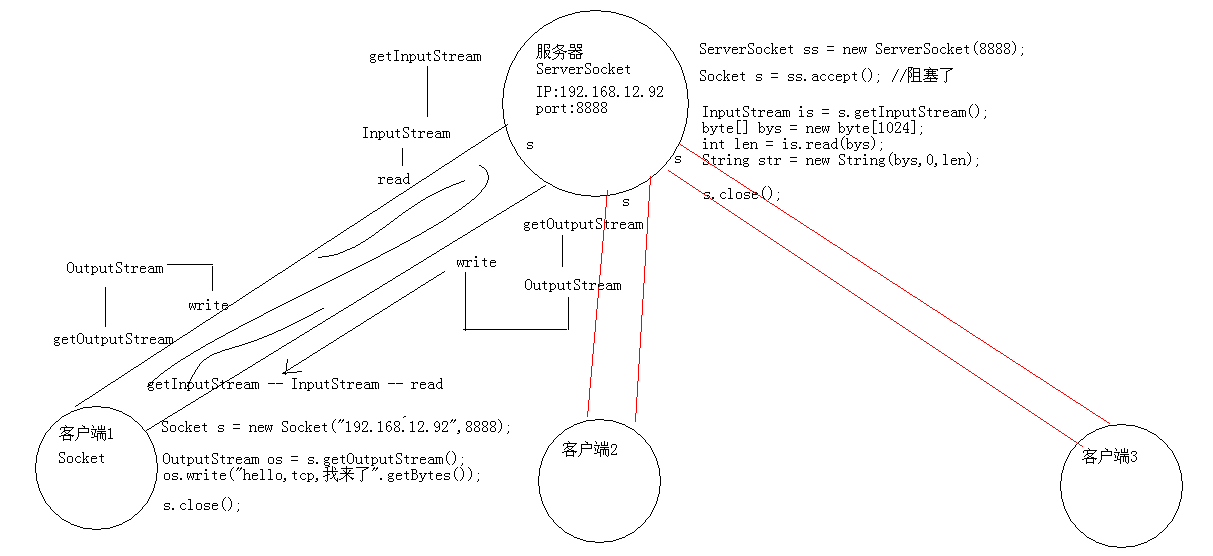
s.close();

// ss.close(); //这个不应该关闭

}

}

## 运行图解



# 列子

## 客户端与服务器端相互通信



## 服务器端无线循环监听

<https://blog.csdn.net/YuDBL/article/details/86024080>

