java的Scoket入门

BIO编程（面向字节流）

BIO编程的基本步骤：

1. 服务器启动一个serverSocket；
2. 客户端启动Socket对服务器进行通信，默认情况下服务器需要对每个客户建立一个线程与之通讯；
3. 客户端发出请求后，先咨询服务器，是否有线程响应，如果没有则会等待，或者被拒绝；
4. 如果有响应，客户端线程会等待请求结束后，再继续执行。

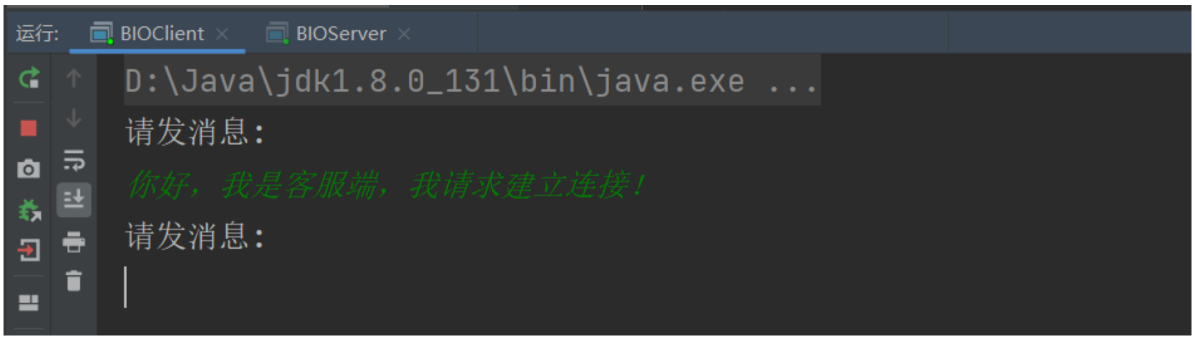
先创建服务端

```
package com.tian.socketproject.BIO.server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * Package: com.tian.socketproject.BIO.server
 * Description:  TODO
 * Author: 田智龙
 * Date: Created in 2021/7/14 19:30
 * Copyright: Copyright (c) 2021
 * Modified By: SmartDragon
 */
//服务端
public class BIOServer {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(1000);
            Socket socket = serverSocket.accept();
            InputStream inputStream = socket.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));
            String msg;
            while ((msg = bufferedReader.readLine())!=null){
                System.out.println("接受到的msg:"+msg);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }finally {

        }
    }
}
```

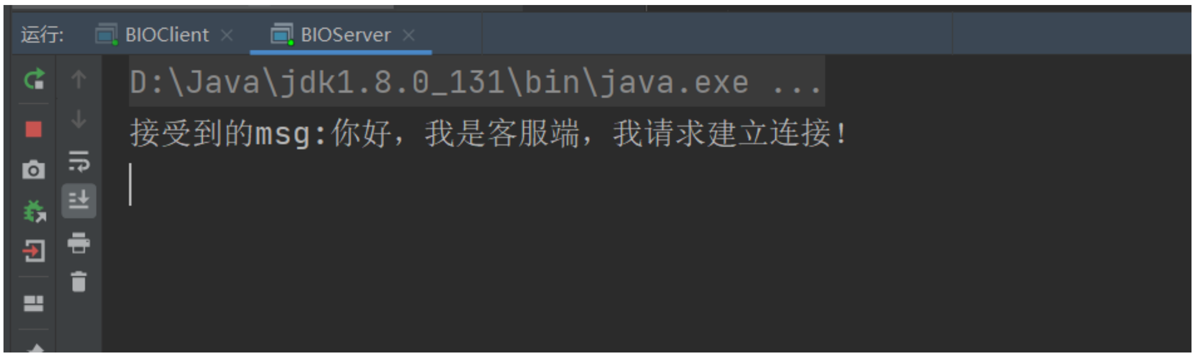
然后创建客户端

```java
package com.tian.socketproject.BIO.client;

import java.io.IOException;
import java.io.PrintStream;
import java.net.Socket;
import java.util.Scanner;

/**
 * Package: com.tian.socketproject.BIO.client
 * Description:  TODO
 * Author: 田智龙
 * Date: Created in 2021/7/14 19:33
 * Copyright: Copyright (c) 2021
 * Modified By: SmartDragon
 */
public class BIOClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost",1000);
            PrintStream printStream = new PrintStream(socket.getOutputStream());
            Scanner scanner = new Scanner(System.in);
            String msg;
            while (true){
                System.out.println("请发消息:");
                msg = scanner.nextLine();
                printStream.println(msg);
                printStream.flush();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

运行结果如下:



```
D:\Java\jdk1.8.0_131\bin\java.exe ...
请发消息:
你好，我是客服端，我请求建立连接！
请发消息:
```



```
D:\Java\jdk1.8.0_131\bin\java.exe ...
接受到的msg:你好，我是客服端，我请求建立连接！
```

BIO的改进（利用线程来实现）

首先创建一个线程类实现服务端的任务

```
package com.tian.socketproject.BIOThread.thread;

import com.tian.socketproject.socket.Server;

import java.io.*;
import java.net.Socket;

/**
 * Package: com.tian.socketproject.BIOThread.thread
 * Description:  TODO
 * Author: 田智龙
 * Date: Created in 2021/7/14 19:38
 * Copyright: Copyright (c) 2021
 * Modified By: SmartDragon
 */
//创建一个服务线程
public class ServerThread extends Thread{

    //套接字
    private Socket socket;

    //构造函数
    public ServerThread(Socket socket){
        this.socket = socket;
    }

    @Override
    public void run(){
        try {
            InputStream inputStream = socket.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));
            String msg;
            while ((msg = bufferedReader.readLine())!=null){
                System.out.println("接受到的消息为: "+msg);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

然后创建一个Server服务端

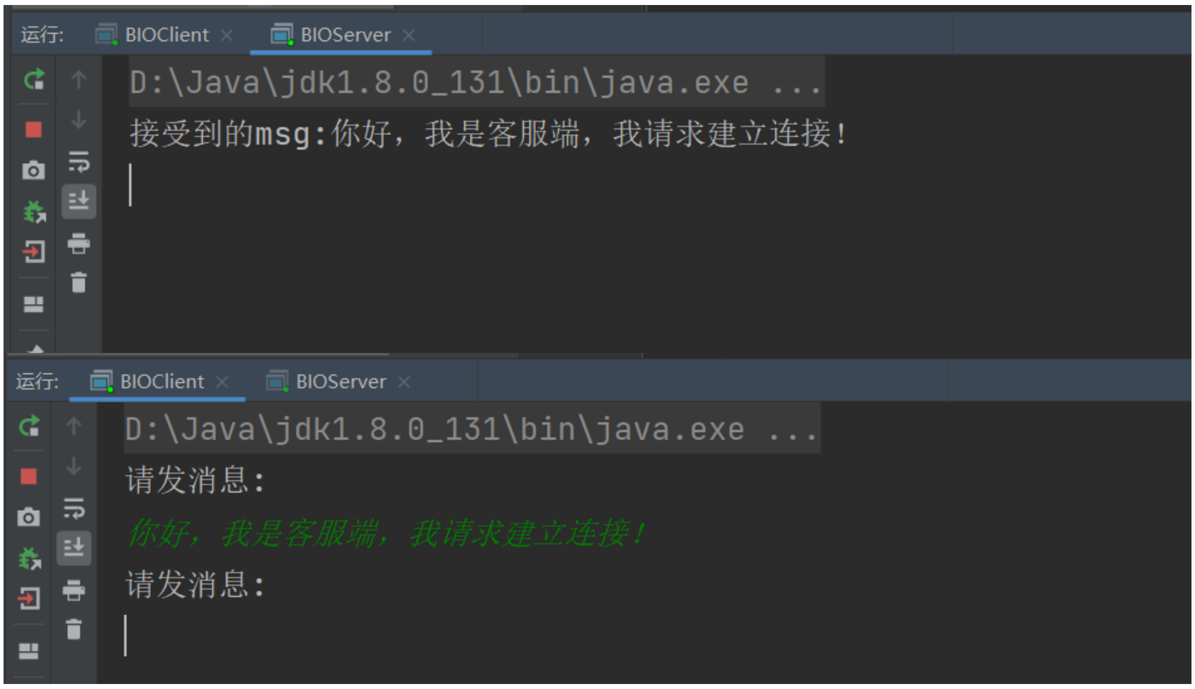
```
package com.tian.socketproject.BIOThread.server;

import com.tian.socketproject.BIOThread.thread.ServerThread;
import jdk.nashorn.internal.runtime.Scope;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
```

```java
/**
 * Package: com.tian.socketproject.BIOThread.server
 * Description:  TODO
 * Author: 田智龙
 * Date: Created in 2021/7/14 19:44
 * Company: 山东理工大学
 * Copyright: Copyright (c) 2021
 * Modified By: SmartDragon
 */
public class Server {
    public static void main(String[] args) {
        try {
            //创建端口号
            ServerSocket serverSocket = new ServerSocket(8888);
            while (true){
                //获取连接
                Socket socket = serverSocket.accept();
                //启动线程
                new ServerThread(socket).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

最后创建客户端

```java
package com.tian.socketproject.BIOThread.client;

import java.io.IOException;
import java.io.PrintStream;
import java.net.Socket;
import java.util.Scanner;

/**
 * Package: com.tian.socketproject.BIOThread.client
 * Description:  TODO
 * Author: 田智龙
 * Date: Created in 2021/7/14 19:49
 * Copyright: Copyright (c) 2021
 * Modified By: SmartDragon
 */
//客户端
public class Client {
    public static synchronized void main(String[] args) {
        try {
            Socket socket = new Socket("localhost",8888);
            PrintStream printStream = new PrintStream(socket.getOutputStream());
            Scanner scanner = new Scanner(System.in);
            String msg;
            while (true){
                System.out.println("请发送消息：");
                msg = scanner.nextLine();
                printStream.println(msg);
                printStream.flush();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }finally {

        }
    }
}
```

运行结果如下(和之间的一样只不过是利用线程来创建服务，简化了而已！):



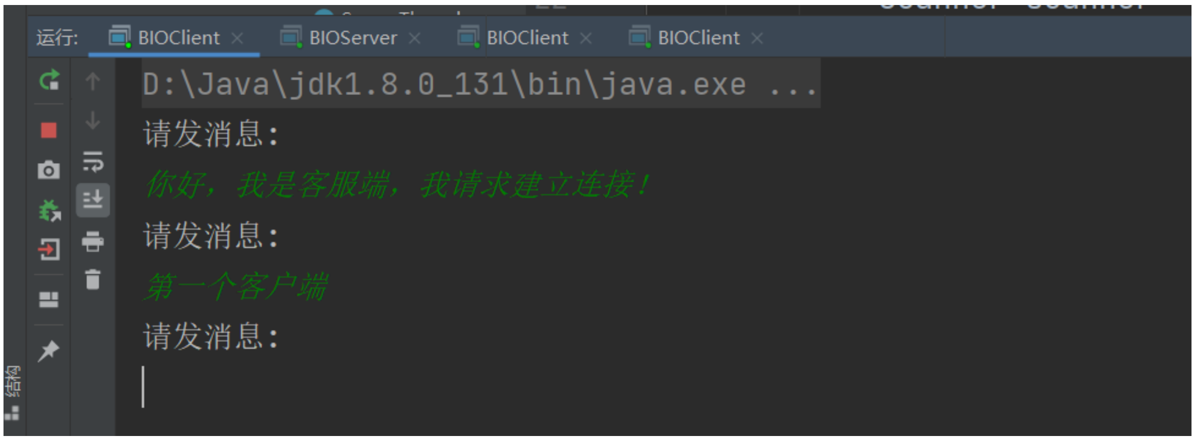
```
D:\Java\jdk1.8.0_131\bin\java.exe ...
接受到的msg:你好，我是客服端，我请求建立连接!
```

```
D:\Java\jdk1.8.0_131\bin\java.exe ...
请发消息:
你好，我是客服端，我请求建立连接!
请发消息:
```

BIO的缺点

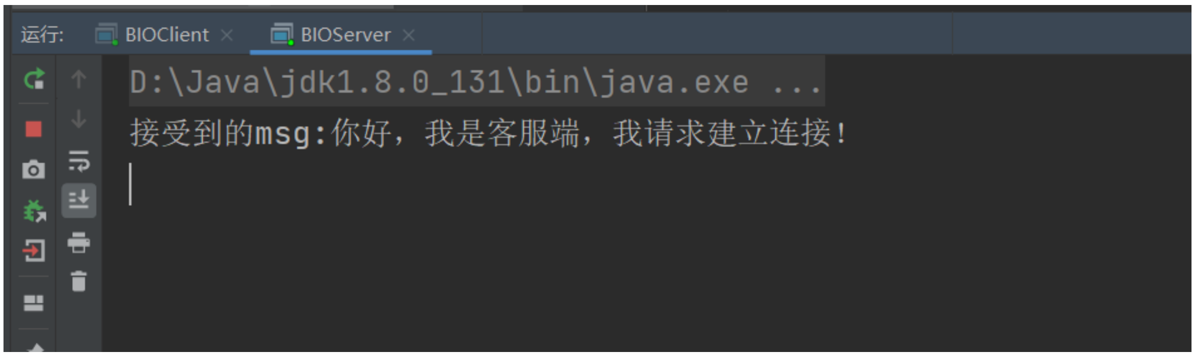
当同时开启多个客户端时，如果同时给服务端发消息，那么只有一个客户端发送的消息服务端可以接受到，而其他客户端发送的消息只能等待，进而服务端接受不到消息！这样大大影响了并发时的效率！于是NIO就来捧场来了！（如下图）

第一个客户端



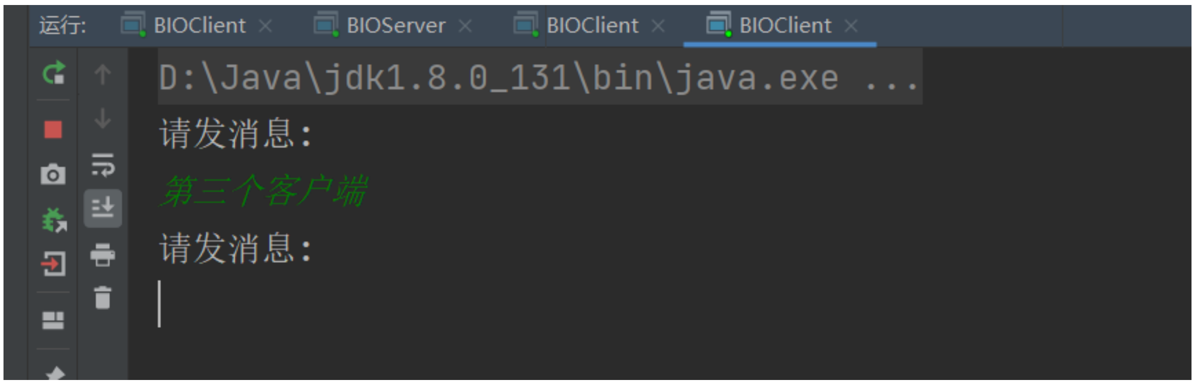
```
D:\Java\jdk1.8.0_131\bin\java.exe ...
请发消息:
你好，我是客服端，我请求建立连接!
请发消息:
第一个客户端
请发消息:
```

第二个客户端



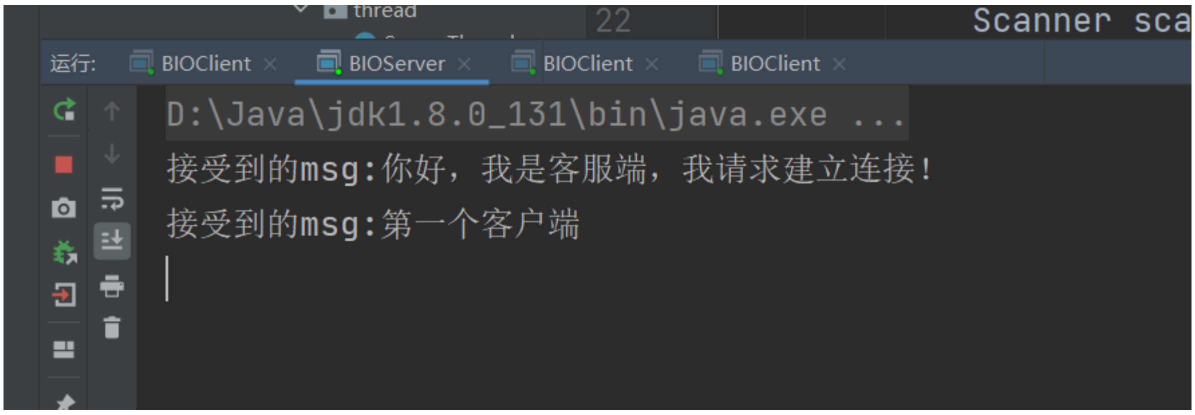
```
D:\Java\jdk1.8.0_131\bin\java.exe ...
接受到的msg:你好，我是客服端，我请求建立连接!
```

第三个客户端



```
D:\Java\jdk1.8.0_131\bin\java.exe ...
请发消息:
第三个客户端
请发消息:
```

服务端只接受第一个客户端发送的消息



```
D:\Java\jdk1.8.0_131\bin\java.exe ...
接受到的msg:你好，我是客服端，我请求建立连接!
接受到的msg:第一个客户端
```

NIO编程（面向管道和选择器的）

NIO编程的基本步骤:

1、创建Selector；

2、创建ServerSocketChannel，并绑定监听端口;

3、将Channel设置为非阻塞模式;

4、将Channel注册到Selector上，监听连接事件;

5、循环调用Selector的select方法，检测就绪情况;

6、调用selectedKeys方法获取就绪channel集合；

7、判断就绪事件种类，调用业务处理方法；

8、根据业务需要决定是否再次注册监听事件，重复执行第3步操作。

创建NIO服务端

```
package com.tian.socketproject.NIO.one;
```

```java
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;
import java.util.Iterator;
/**
 * Package: com.tian.socketproject.BIOThread.thread
 * Description:  TODO
 * Author: 田智龙
 * Date: Created in 2021/7/14 19:38
 * Copyright: Copyright (c) 2021
 * Modified By: SmartDragon
 */
/**
    目标：NIO非阻塞通信下的入门案例：服务端开发
 */
public class Server {
    public static void main(String[] args) throws Exception {
        System.out.println("----服务端启动---");
        // 1、获取通道
        ServerSocketChannel ssChannel = ServerSocketChannel.open();
        // 2、切换为非阻塞模式
        ssChannel.configureBlocking(false);
        // 3、绑定连接的端口
        ssChannel.bind(new InetSocketAddress(9999));
        // 4、获取选择器Selector
        Selector selector = Selector.open();
        // 5、将通道都注册到选择器上去，并且开始指定监听接收事件
        ssChannel.register(selector , SelectionKey.OP_ACCEPT);
        // 6、使用Selector选择器轮询已经就绪好的事件
        while (selector.select() > 0){
            System.out.println("开始一轮事件处理~~~");
            // 7、获取选择器中的所有注册的通道中已经就绪好的事件
            Iterator<SelectionKey> it = selector.selectedKeys().iterator();
            // 8、开始遍历这些准备好的事件
            while (it.hasNext()){
                // 提取当前这个事件
                SelectionKey sk = it.next();
                // 9、判断这个事件具体是什么
                if(sk.isAcceptable()){
                    // 10、直接获取当前接入的客户端通道
                    SocketChannel schannel = ssChannel.accept();
                    // 11 、切换成非阻塞模式
                    schannel.configureBlocking(false);
                    // 12、将本客户端通道注册到选择器
                    schannel.register(selector , SelectionKey.OP_READ);
                }else if(sk.isReadable()){
                    // 13、获取当前选择器上的读就绪事件
                    SocketChannel sChannel = (SocketChannel) sk.channel();
                    // 14、读取数据
                    ByteBuffer buf = ByteBuffer.allocate(1024);
                    int len = 0;
                    while((len = sChannel.read(buf)) > 0){
                        buf.flip();
                        System.out.println(new String(buf.array() , 0, len));
                        buf.clear();// 清除之前的数据
```

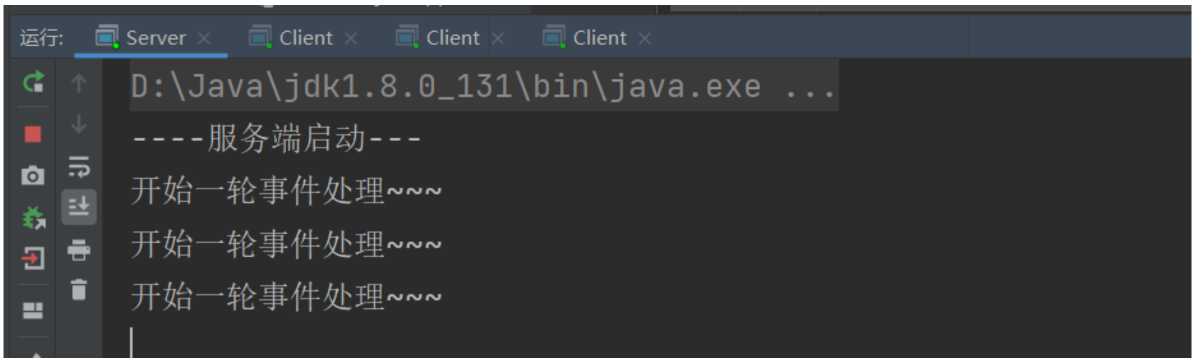
```
                    }
                }

                it.remove(); // 处理完毕之后需要移除当前事件
            }
        }
    }
}
```

创建客户端

```
package com.tian.socketproject.NIO.one;

import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SocketChannel;
import java.util.Scanner;

/**
 * Package: com.tian.socketproject.BIOThread.thread
 * Description:  TODO
 * Author: 田智龙
 * Date: Created in 2021/7/14 19:38
 * Copyright: Copyright (c) 2021
 * Modified By: SmartDragon
 */

/**
    目标：客户端案例实现-基于NIO非阻塞通信。
 */
public class Client {
    public static void main(String[] args) throws Exception {
        // 1、获取通道
        SocketChannel sChannel = SocketChannel.open(new
InetSocketAddress("127.0.0.1", 9999));
        // 2、切换成非阻塞模式
        sChannel.configureBlocking(false);
        // 3、分配指定缓冲区大小
        ByteBuffer buf = ByteBuffer.allocate(1024);
        // 4、发送数据给服务端
        Scanner sc = new Scanner(System.in);
        while (true){
            System.out.println("请说：");
            String msg = sc.nextLine();
            buf.put(("波妞："+msg).getBytes());
            buf.flip();
            sChannel.write(buf);
            buf.clear();
        }
    }
}
```

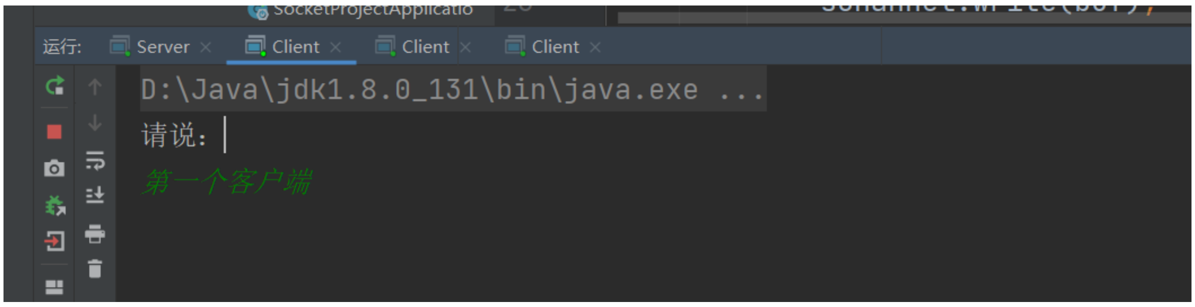
运行结果:

服务器启动:



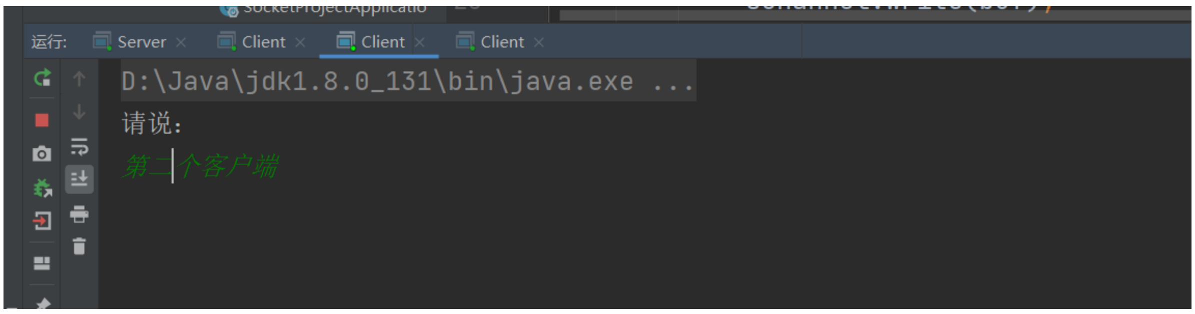
```
D:\Java\jdk1.8.0_131\bin\java.exe ...
----服务端启动---
开始一轮事件处理~~~
开始一轮事件处理~~~
开始一轮事件处理~~~
```

第一个客户端:



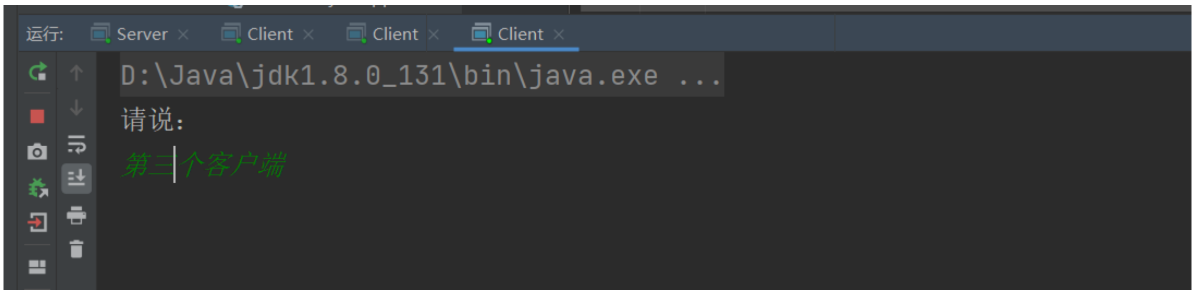
```
D:\Java\jdk1.8.0_131\bin\java.exe ...
请说:
第一个客户端
```

第二个客户端:



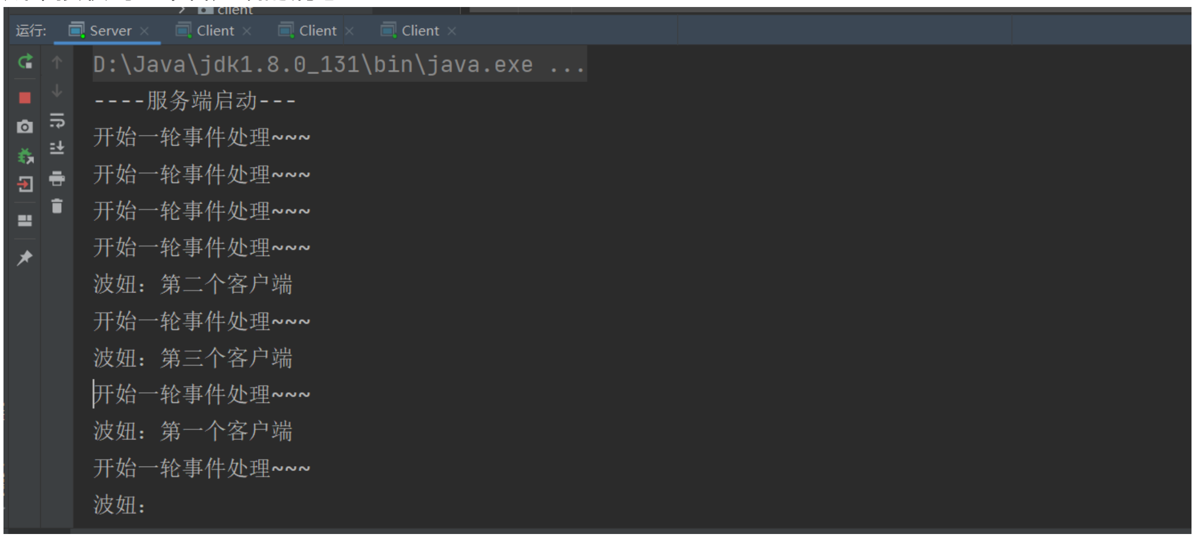
```
D:\Java\jdk1.8.0_131\bin\java.exe ...
请说:
第二个客户端
```

第三个客户端:



```
D:\Java\jdk1.8.0_131\bin\java.exe ...
请说:
第三个客户端
```

成果接收到三个客户端的消息!



```
D:\Java\jdk1.8.0_131\bin\java.exe ...
----服务端启动---
开始一轮事件处理~~~
开始一轮事件处理~~~
开始一轮事件处理~~~
开始一轮事件处理~~~
波妞: 第二个客户端
开始一轮事件处理~~~
波妞: 第三个客户端
开始一轮事件处理~~~
波妞: 第一个客户端
开始一轮事件处理~~~
波妞:
```

大功告成!

作者：SmartDragon

QQ邮箱：2455404279@qq.com

欢迎留言交流哈！你的留言就是我的进步啊!