

C#Socket基于tcp的简单聊天程序

C#Socket编程

一、简单了解服务端和客户端各自的功能。

首先应该清楚服务端（Server）和客户端（Client）它们各自的功能。

（1）服务端（Server）：

负责接收客户端的请求，然后根据客户端请求的内容不同而给客户端返回相应的数据。

（2）客户端（Client）：

链接服务端，向服务端发送自己的业务需求（也就是数据），然后接受服务端返回过来的信息。

（3）分析服务端和客户端的功能，可以很清楚的知道，它们完成了数据之间的交流，或者说是业务之间的相互传递与获取。

二、服务器与客户端之间信息传递的桥梁（Socket）

（1）服务器和客户端进行信息传递的通道，socket套接字分为很多种类型，它是一个协议族，常用的协议TCP/IP和UDP两种。

（2）简单来说就是通过socket协议能够进行通信，每种编程语言socket的写法都八九不离十，创建socket通信的步骤都十分接近。

（3）服务端socket和客户端socket通过对方的IP地址和对应应用程序的PORT（端口号）进行连接和数据传输。

三、C#中创建socket的一般方式以及大致业务流程

（1）首先定义一个Socket套接字对象(这里协议族的类型我们选择TCP协议，TCP传输数据时安全、稳定、可靠，当然对应的性能比UDP差)

```
Socket socket_server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
```

（2）不管是客户端还是服务端，创建socket套接字的方式都一样，但是服务端（Server需要绑定该服务端的IP和端口号，以便让客户端进行连接），为了方便这里我绑定的是本地的IP（localhost）

```
IPAddress ipAddress = IPAddress.Parse("127.0.0.1");
```

```
int port = 8888;
```

（3）为服务端绑定IP和端口

```
socket_server.Bind(new IPEndPoint(ipAddress, port));
```

(4) 绑定好IP和PORT后就开始监听 (将创建的套接字变为监听套接字,以及设置最大同时监听数量)

```
socket_server.Listen(100);
```

(5) 监听完毕就可以等待客户端的连接了(注意:如果没有客户端连接到来,那么Accept()方法会一直处于阻塞状态,会阻塞当前线程,直到有连接到来,线程才会接阻塞,然后将客户端的套接字储存下来,以便接下来的数据传输使用。)

```
Socket clientSocket = socket_server.Accept();
```

(6) 接受客户端的消息 (使用我们上一步储存的客户端套接字来接收)

//这里接受消息的方式是以字节流来接收的,所有用一个byte类型的数组来储存,Receive方法还返回接受直接的实际长度,Encoding.UTF8.GetString()方法将byte类型的数据转化为字符串

```
byte[] strbyte = new byte[1024];
int count = clientsocket.Receive(strbyte);
string ret = Encoding.UTF8.GetString(strbyte, 0, count)
```

(7) 向客户端发送消息 (发送消息使用Send方法进行)

```
Console.WriteLine("请输入要发送的消息:");
string str = Console.ReadLine();
byte[] strbyte = Encoding.UTF8.GetBytes(str);
clientsocket.Send(strbyte);
```

四、完整的服务端代码如下：



```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace SocketServer
{
    /// <summary>
    /// 服务端
    /// </summary>
    class Serversocket
    {
        /// <summary>
        /// 服务端入口
        /// </summary>
        private static Serversocket socketserver;
        public static Serversocket Instance()
        {
            if (socketserver == null)
            {
                socketserver = new Serversocket();
            }
        }
    }
}
```

```
        return socketserver;
    }

    private Socket server;
    private IPAddress ipAddress;
    private int port;

    /// <summary>
    /// 初始化数据
    /// </summary>
    public void Init()
    {
        ipAddress = IPAddress.Parse("127.0.0.1");
        port = 8888;
        CreateSocket();
        BindAndListen();
        WaitClientConnection();
    }

    /// <summary>
    /// 1.创建服务端的套接字
    /// </summary>
    private void CreateSocket()
    {
        server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

    }
    /// <summary>
    /// 2.绑定服务器ip和端口
    /// </summary>
    private void BindAndListen()
    {
        server.Bind(new IPEndPoint(ipAddress, port));
        server.Listen(100);
    }

    /// <summary>
    /// 用一个列表储存连接成功的客户端
    /// </summary>
    private List<Socket> ConnectSocketList = new List<Socket>();
    /// <summary>
    /// 用一个字典储存所有接受客户端消息以及发送给客户端消息的线程
    /// </summary>
    private Dictionary<int, List<Thread>> recvThreadList = new Dictionary<int,
List<Thread>>();

    /// <summary>
    /// 3.等待客户端的连接
    /// </summary>
    private void WaitClientConnection()
    {
        int index = 1;
        while (true)
        {
            Console.WriteLine("当前链接数量：" + recvThreadList.Count);
            Console.WriteLine("等待客户端的连接：");
            Socket ClientSocket = server.Accept();
```

```

        if (ClientSocket!=null)
        {
            Console.WriteLine("{0}连接成功!", ClientSocket.RemoteEndPoint);
            ConnectSocketList.Add(ClientSocket);
            //创建接受客户端消息的线程, 并将其启动
            Thread recv = new Thread(RecvMessage);
            recv.Start(new ArrayList { index, ClientSocket });
            Thread send = new Thread(SendMessage);
            send.Start(new ArrayList { index, ClientSocket });
            recvThreadList.Add(index, new List<Thread> { recv, send });
            index++;
        }
    }

    /// <summary>
    /// 接受客户端的消息
    /// </summary>
    /// <param name="clientsocket"></param>
    private void RecvMessage(object client_socket)
    {
        ArrayList arraylist = client_socket as ArrayList;
        int index = (int)arraylist[0];
        Socket clientsocket = arraylist[1] as Socket;
        while (true)
        {
            try
            {
                byte[] strbyte = new byte[1024];
                int count = clientsocket.Receive(strbyte);
                string ret = Encoding.UTF8.GetString(strbyte, 0, count);
                Console.WriteLine("{0}给你发送了消息 :
{1}", clientsocket.RemoteEndPoint, ret);
            }
            catch (Exception)
            {
                //客户端离去时终止线程
                Console.WriteLine("代号为:{0}的客户端已经离去!", index);
                recvThreadList[index][0].Abort();
            }
        }
    }

    /// <summary>
    /// 向客户端返回消息
    /// </summary>
    private void SendMessage(object client_socket)
    {
        ArrayList arraylist = client_socket as ArrayList;
        int index = (int)arraylist[0];
        Socket clientsocket = arraylist[1] as Socket;
        while (true)
        {
            try
            {
                Console.WriteLine("请输入要发送的消息:");
                string str = Console.ReadLine();
                byte[] strbyte = Encoding.UTF8.GetBytes(str);
            }
        }
    }

```

```
        clientsocket.Send(strbyte);
    }
    catch (Exception)
    {
        Console.WriteLine("代号为:{0}的客户端已经离去!消息发送失败!");
        recvThreadList[index][1].Abort();
        recvThreadList.Remove(index);
    }
}
}
}
}
```



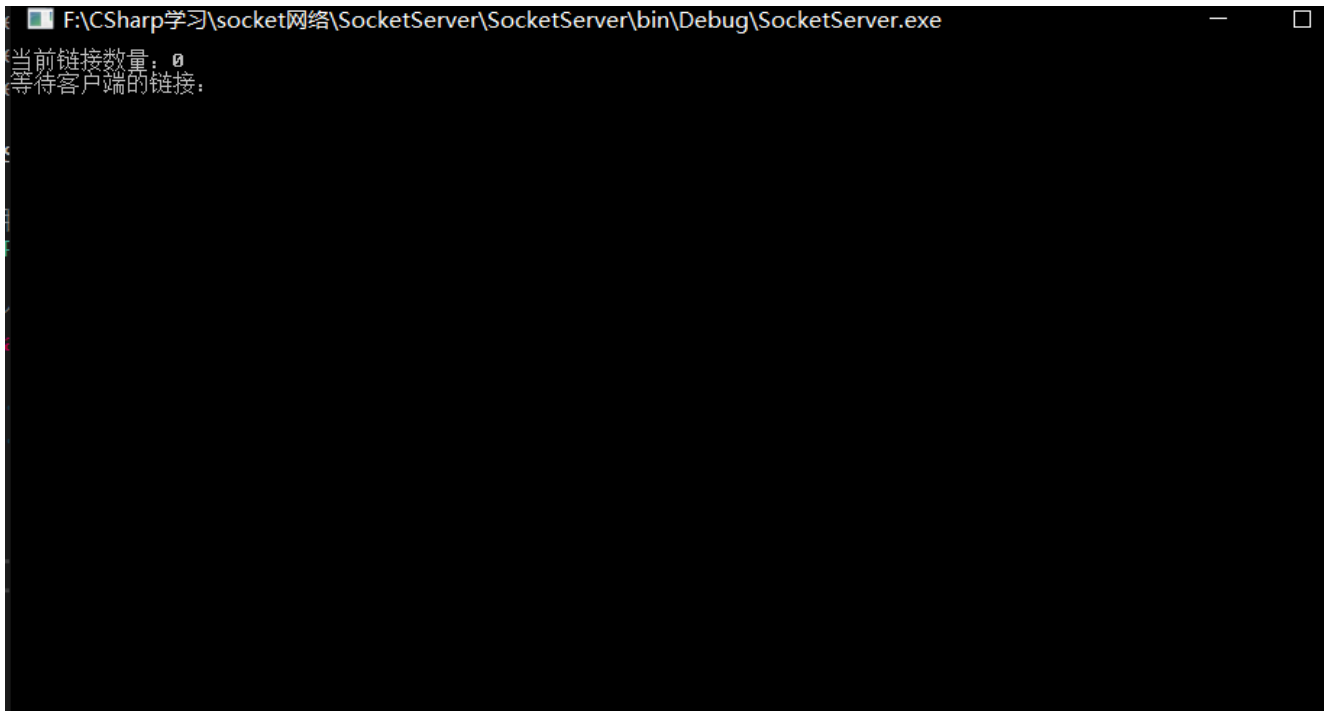
启动服务器的代码：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SocketServer
{
    class Program
    {
        static void Main(string[] args)
        {
            Serversocket.Instance().Init();
            Console.ReadKey();
        }
    }
}
```



启动后的运行效果：



五、上面的服务端代码允许多个客户端同时访问，而且能对不同的客户端分开处理主要应用到了C#中的Thead模块

(1) 创建一条线程

```
Thread t1 = new Thread(对应要执行的函数)
//Thread方法有四个重载，常用的三个
public Thread(ThreadStart start); //接受一个无参数的且无返回值的委托
public Thread(ParameterizedThreadStart start); //接受一个带object类型参数的且无返回值的委托
public Thread(ThreadStart start, int maxStackSize); //接受一个无参数的且无返回值的委托, 可以指定线程最大堆栈大小
```

(2) 启动一条线程

```
t1.start() //可以指定object类型的参数
```

(3) 终止一条线程

```
t1.Abort()
```

(4) 如果不用多线程去处理单个客户端的消息发送和接受，那么当有多个客户端连接时，如果其中一个客户端发生阻塞，那么后面的所有客户端都将阻塞。

六、简单客户端代码：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace Socket_Client_02
{
```

```
/// <summary>
/// 客户端
/// </summary>
class Socket_Client
{
    public Socket_Client(string ip,int port)
    {
        this.IPAddress = IPAddress.Parse(ip);
        this.port = port;
    }

    /// <summary>
    /// 套接字
    /// </summary>
    private Socket client_socket;
    /// <summary>
    /// 客户端要连接的ip地址
    /// </summary>
    private IPAddress ipAddress;
    /// <summary>
    /// 客户端要连接的端口号
    /// </summary>
    private int port;
    /// <summary>
    /// 创建客户端连接的套接字
    /// </summary>
    /// <returns></returns>
    public Socket Create_Client_Socket()
    {
        return new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
    }
    /// <summary>
    /// 连接服务器
    /// </summary>
    public void Connect_Server()
    {
        client_socket = Create_Client_Socket();
        //tcp连接服务器的时候只需要连接一次，因为tcp是长链接
        client_socket.Connect(new IPEndPoint(ipAddress, port));
    }

    /// <summary>
    /// 接收来自服务器的消息
    /// </summary>
    public void Recv_Msg_By_Client()
    {
        while (true)
        {
            byte[] ser_msg = new byte[1024];
            int count = client_socket.Receive(ser_msg);
            string str_msg = Encoding.UTF8.GetString(ser_msg, 0, count);
            if (count > 0)
            {
                Console.WriteLine("接收到来自{0}的消息为：{1}",
client_socket.RemoteEndPoint, str_msg);
            }
        }
    }
}
```

```
    }

    /// <summary>
    /// 向服务器发送请求
    /// </summary>
    public void Request_Client()
    {
        while (true)
        {
            Console.WriteLine("请输入你要发送到服务器的消息：");
            string send_msg = Console.ReadLine();
            byte[] by_msg = Encoding.UTF8.GetBytes(send_msg);
            client_socket.Send(by_msg);
        }
    }
}
```



客户端启动代码：




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Socket_Client_02
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("请输入你要连接的服务器的ip地址：");
            string ip = Console.ReadLine();
            Console.WriteLine("请输入你要连接的服务器的端口号：");
            int port = int.Parse(Console.ReadLine());

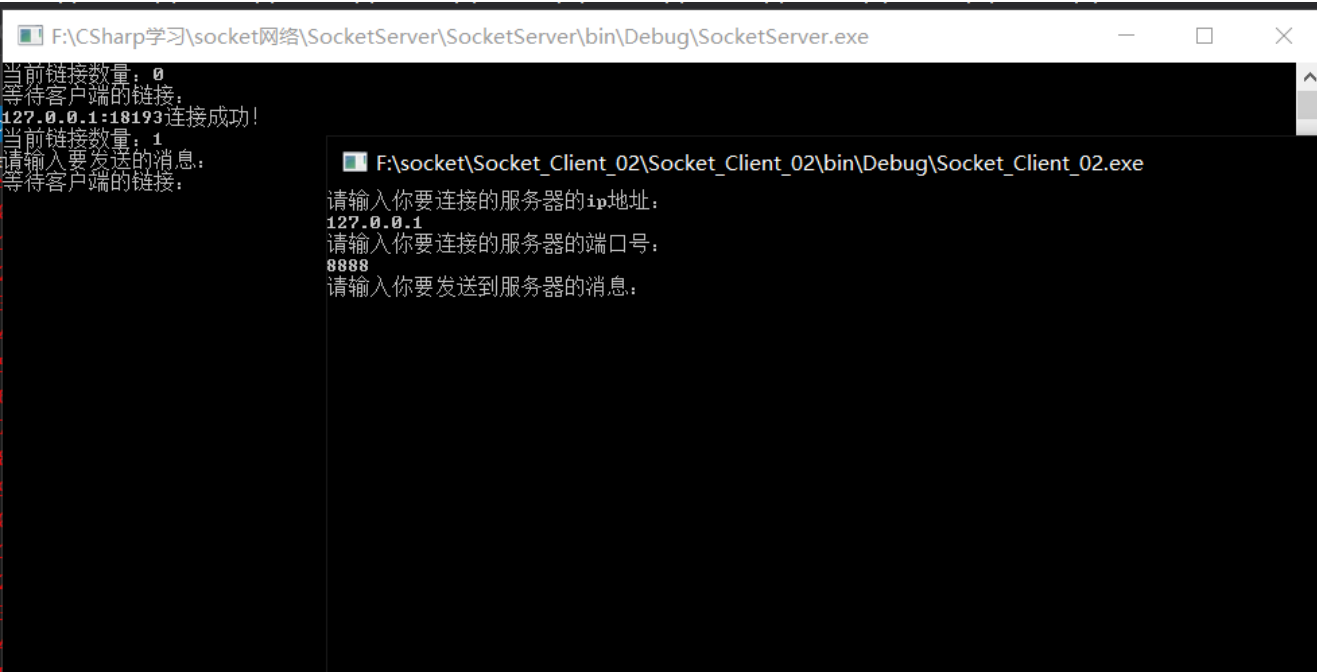
            //创建套接字
            Socket_Client s = new Socket_Client(ip, port);
            //连接服务器
            s.Connect_Server();
            //接收服务器的消息
            Thread recv = new Thread(s.Recv_Msg_By_Client);
            //给服务器发送消息
            Thread send = new Thread(s.Request_Client);
            recv.Start();
            send.Start();

            recv.Join();
            send.Join();
            Console.ReadKey();
        }
    }
}
```



```
}  
}  

```

运行效果图：



先开启服务端，然后用客户端去连接，连接之后就能相互聊天了。

分类: [C# Socket编程](#)

标签: [C#socket编程服务端多线程](#)

好文要顶

关注我

收藏该文







[一支梨花压海棠](#)
[关注 - 1](#)
[粉丝 - 0](#)
[+加关注](#)

1

0

» 下一篇：[python之socket简易聊天器](#)

posted @ 2019-08-25 14:16

一支梨花压海棠

阅读(3142)

评论(3)

编辑

收藏

评论

#1楼 2019-09-16 18:57 | Mr_Zhanx

博主流弊 支持一个

支持(0)

反对(0)

#2楼 2020-03-11 17:12 | Dindy-R

博主，若果要让两个客户端聊天呢（本人不是通讯专业的，很好奇）

#3楼 2020-03-12 23:43 | s_xiao_wang

发现仨bug

俩非常严重的bug

- telnet出一个空连接给服务器之后，握手失败，网络堵死
- 如果多个客户端和服务端链接，服务端的发报只能顺序发放，A一次B一次，无法选择
- 如果一个客户端断开连接，服务端崩溃

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，访问 [网站首页](#)。

【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【活动】腾讯云服务器推出云产品采购季 1核2G首年仅需99元

【推荐】精品问答：大数据常见问题之 flink 五十问

【推荐】独家下载 | 《大数据工程师必读手册》揭秘阿里如何玩转大数据

相关博文：

- [利用TCP和UDP协议，实现基于Socket的小聊天程序\(初级版\)](#)
- [利用TCP传输协议实现基于Socket的聊天程序\(高级版_多线程\)](#)
- [基于C#的socket编程的TCP同步实现](#)
- [聊天程序（基于Socket、Thread）](#)
- [基于Tcp协议的简单Socket通信实例（JAVA）](#)
- » [更多推荐...](#)

[合辑](#) | [学习python不可不知的开发者词条汇总！](#)

最新 IT 新闻：

- [三大运营商及中国铁塔今年将投近2000亿元建设5G](#)
- [全新 Powerbeats 体验：它就像你们想要的 iPhone 9](#)
- [宝宝树CTO詹宏勇已离职，乐一帆接任](#)
- [Redmi发布智能电视MAX 98：98英寸面板售价19999元](#)
- [盖茨公开信谈新冠病毒：不管我们多伟大，病毒都会让世界陷入停滞](#)
- » [更多新闻...](#)