C# 网络编程之使用Socket类Send、Receive方法的同步通讯

经过几天学习，终于解决了再C#网络编程中使用Socket类Send和Receive方法开发的客户端和服务端的同步通讯程序；实现了又客户端想服务器发送消息的界面程序.主要使用的方法是：

1.Socket套接字编程的知识,通过IPAddress定义一个IP地址,IPEndPoint定义一个主机,Socket实例套接字对象sock和线程Thread的的成员变量;

2.再调用方法bind绑定端口、listen监听端口、accept接受连接请求、connect请求连接来连接客户端和服务器;

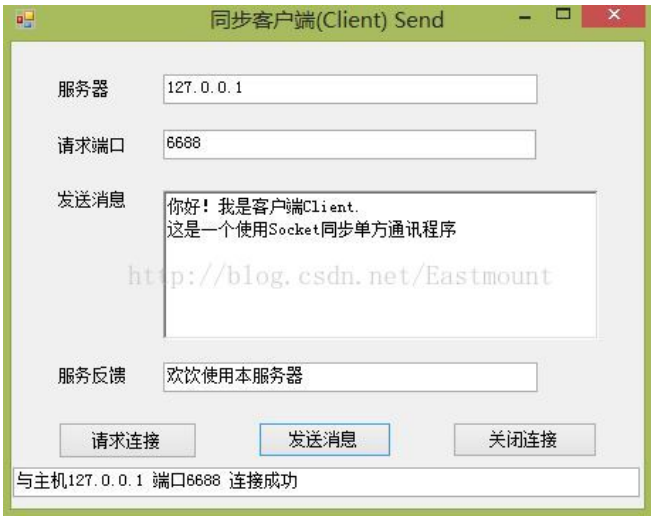
3.建立连接后通过Send和Receive方法通过线程循环接受连接请求中发送的消息,实现通信并显示在相应的控件中;

4.最后调用socket的close和shutdown方法关闭套接字,停止连接监听.

下面是程序运行后的结果:

(服务端接受客户端发送的消息:这是一个单方的通信,但实现双方的方法相同,因为服务端的"欢饮使用本服务器"也反馈显示在了客户端)

(客户端)



下面是本程序的源代码：

服务端

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

//添加新的命名空间
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace tbServer
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        //添加私有成员
        private IPAddress myIP = IPAddress.Parse("127.0.0.1");   //定义IP对象
        private IPEndPoint MyServer;                             //定义主机
        private Socket sock;                                     //套接字对象实例
        private bool sign = true;                                //控制循环
        private Thread thread;                                   //创建控制线程
        private Socket socklin;                                  //临时套接字,接受客户端连接请求

        //双击"开始监听"按钮添加Click事件
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                myIP = IPAddress.Parse(textBox1.Text);           //字符串转换为IP
            }
            catch
            {
                MessageBox.Show("你输入的IP地址格式错误!");
            }

            try
            {
                //定义主机
                MyServer = new IPEndPoint(myIP, Int32.Parse(textBox2.Text));
                //构造套接字
                sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
                //绑定端口
                sock.Bind(MyServer);
                //开始监听
                sock.Listen(10);
                //状态栏信息添加 textBox3替代statusStrip1(不会用)
                textBox3.Text = "主机" + textBox1.Text + " 端口" + textBox2.Text + " 开始监听";
                //构造线程
                thread = new Thread(new ThreadStart(targett)); //targett自定义函数:接受客户端连接请求
                //启动线程用于接受连接和接受数据
                thread.Start();
```

```
    }            catch(Exception msg) {
        textBox3.Text = msg.Message;
    }
}

//targett():自定义函数,该方法循环开始接受客户端的连接请求
private void targett()
{
    socklin = sock.Accept();   //接受连接请求
    sign = true;               //循环标志变量true

    //连接
    if (socklin.Connected)
    {
        textBox3.Text = "与客户端连接";

        //信息反馈给客户端Client
        Byte[] byteNum = new Byte[64];                      //构造字节数组
        byteNum = System.Text.Encoding.BigEndianUnicode.GetBytes("欢饮使用本服务器".ToCharArray());
        socklin.Send(byteNum,byteNum.Length,0);             //发送数据

        //sign为true 循环接受数据
        while (sign)
        {
            Byte[] byteNum2 = new Byte[128];
            socklin.Receive(byteNum2,byteNum2.Length,0);   //接受数据
            string str = System.Text.Encoding.BigEndianUnicode.GetString(byteNum2);
            richTextBox1.AppendText(str+"\r\n");           //显示字符串

            //获取richTextBox1行数
            int length = richTextBox1.Lines.Length;
            //如果客户端发送倒数第二行的字符串为"@@@" 断开连接
            if(richTextBox1.Lines[length-2]=="@@@")
            {
                textBox3.Text = "与客户端断开连接";
                //关闭套接字实例（both表示发送和接受关闭）
                socklin.Shutdown(System.Net.Sockets.SocketShutdown.Both);
                socklin.Close();
                sign = false;                //设为false退出循环
            }
        }
    }
}

//双击"停止监听"按钮添加Click事件
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        sign = false;
        sock.Close();
        textBox3.Text = "主机" + textBox1.Text + "端口" + textBox2.Text + "监听停止";
    }
    catch
    {
        MessageBox.Show("监听尚未开始,关闭无效!");
    }

}

//载入Form是设置非安全访问,防止线程无效操作
```

```
        private void Form1_Load(object sender, EventArgs e)                    {
            //非安全线程访问,不检查线程是否安全
            Control.CheckForIllegalCrossThreadCalls = false;
        }
    }
}
```

客户端

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

//添加新的命名空间
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace tbClient
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        //添加私有成员
        private IPAddress myIP = IPAddress.Parse("127.0.0.1");   //定义IP对象
        private IPEndPoint MyServer;                             //定义主机
        private Socket sock;                                     //套接字对象实例
        private Thread thread;                                   //创建控制线程

        //双击"请求连接"按钮添加Click事件
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                myIP = IPAddress.Parse(textBox1.Text);           //字符串转换为IP
            }
            catch
            {
                MessageBox.Show("你输入的IP地址格式错误!");
            }

            try
            {
                //构造主机
                MyServer = new IPEndPoint(myIP, Int32.Parse(textBox2.Text));
                //构造套接字
                sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
                //请求连接
                sock.Connect(MyServer);
```

```
        //构造线程
        thread = new Thread(new ThreadStart(targett)); //targett自定义函数:接受客户端连接请求
        //启动线程用于接受连接和接受数据                    thread.Start();
        //输出信息
        textBox4.Text = "与主机" + textBox1.Text + " 端口" + textBox2.Text + " 连接成功";
    }
    catch(Exception msg)
    {
        MessageBox.Show(msg.Message);
    }
}

//targett():自定义函数,该方法循环开始接受客户端的连接请求
private void targett()
{
    //构造字节数组
    Byte[] byteNum = new Byte[64];
    //接受数据
    sock.Receive(byteNum, byteNum.Length, 0);
    //将字符数组转换为字符串
    string str = System.Text.Encoding.BigEndianUnicode.GetString(byteNum);
    textBox3.Text = str;
}

//双击"发送消息"按钮添加Click事件
private void button2_Click(object sender, EventArgs e)
{
    //构造字节数组
    Byte[] byteNum = new Byte[64];
    //发送内容
    string send = richTextBox1.Text + "\r\n";
    //将字符串转换为字节数组
    byteNum = System.Text.Encoding.BigEndianUnicode.GetBytes(send.ToCharArray());
    //发送数据
    sock.Send(byteNum,byteNum.Length,0);
    //构造线程
    Thread threadSend = new Thread(new ThreadStart(targett));
    //启动线程接受数据
    threadSend.Start();
}

//双击"关闭连接"按钮添加Click事件
private void button3_Click(object sender, EventArgs e)
{
    Byte[] byteNum = new Byte[64];
    string send = "@@@" + "\r\n";
    byteNum = System.Text.Encoding.BigEndianUnicode.GetBytes(send.ToCharArray());
    sock.Send(byteNum, byteNum.Length, 0);   //将"@@@"发送给服务器

    try
    {
        sock.Close();
        textBox4.Text="主机" + textBox1.Text + "端口" + textBox2.Text + "断开连接";
    }
    catch
    {
        MessageBox.Show("连接尚未建立,断开无效!");
    }
}

//载入Form是设置非安全访问,防止线程无效操作
```

```
        private void Form1_Load(object sender, EventArgs e)
                                                                {
            //非安全线程访问,不检查线程是否安全
            Control.CheckForIllegalCrossThreadCalls = false;
        }
    }
}
```

该程序中我遇到的几个主要问题及解决方法如下:

1.程序初期总是很卡,出现多次未响应情况?

因为socket的Accept()函数是阻塞模式,它的执行会造成程序的阻塞,应该把它放置到线程中执行,否则会阻塞当前线程,出现卡死状态不响应消息,后续代码也不会执行,所以需要把accept放到创建的线程thread中,放入targett()函数中的“socklin = sock.accept()”即可实现;

2.在定义的socket对象实例中sock与socklin(临时接受客户端连接请求)中混淆?

socklin = sock.accept,它就是客户端发送连接的请求,因此在判断连接时是if(socklin.Connected),同时使用socklin的send和receive方法发送和接受数据;

3.总是出现“线程间操作无效:从不是创建控件的线程访问它”的错误?

因为windows窗体控件不是线程安全的,如果几个线程操作某一控件的状态,可能会使该控件的状态不一致,出现争用或死锁状态.我采用的解决方法是添加Form的载入load事件,在load时将CheckForIllegalCrossThreadCalls 属性的值设置为false .这样进行非安全线程访问时,运行环境就不去检验它是否是线程安全的.这是来自与该博客,详细情况见:http://blog.csdn.net/wangchao0605/article/details/5010864

总结:

最后经过一星期的学习与查阅资料,还是把这个程序弄出来了,也学到了很多东西,同时感谢上面的博主和一些书籍.希望这篇文章对大家有用,有错或不足之处见谅!

(BY:Eastmount 2013-7-22 http://blog.csdn.net/eastmount/)