TCP/IP Chat Application Using C#

ALLEXY, 30 Jan 2006 CPOL

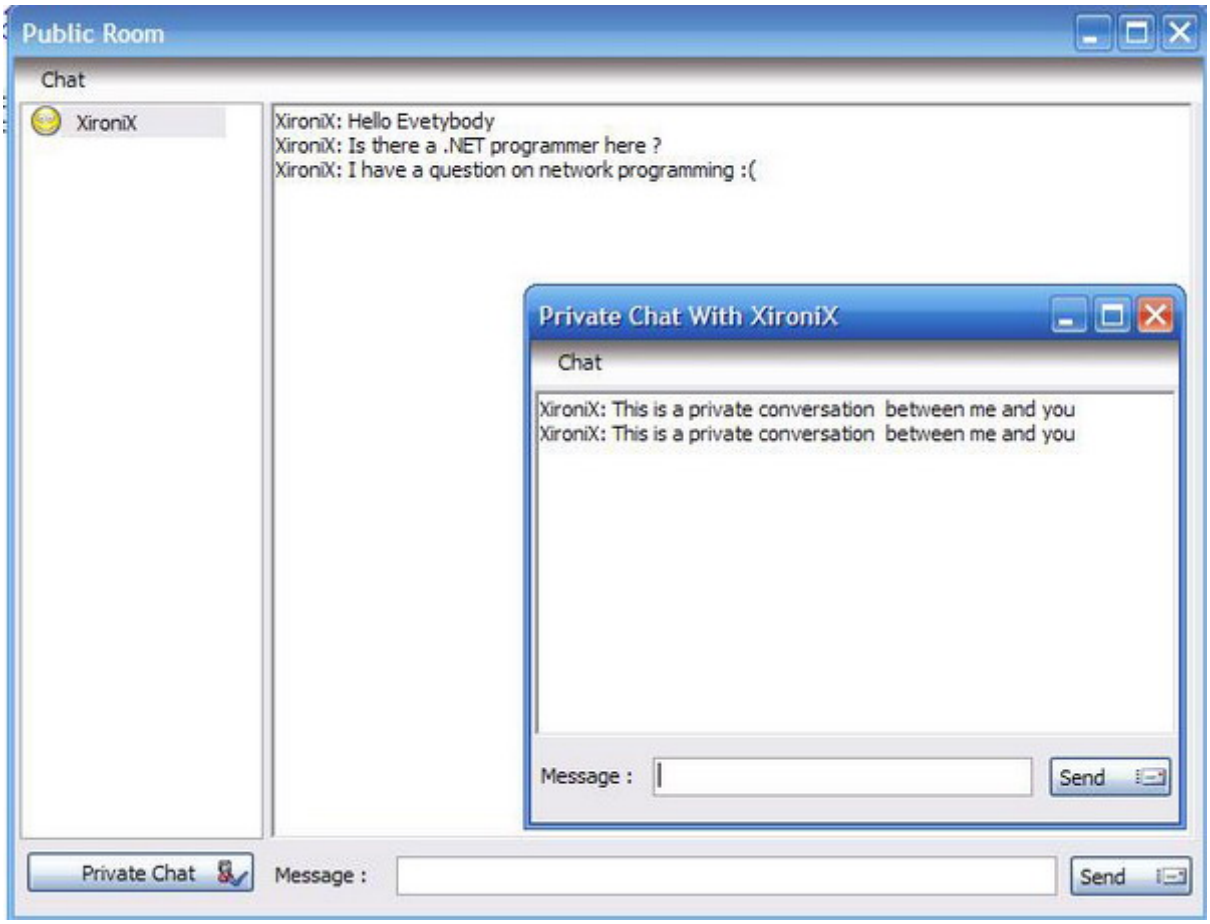
4.86 (199 votes)

Rate this:

This is a LAN chat application with TCP/IP socket programming technology in C#. This application is a multi thread network application and works in a non-blocking way. Public and private chat is also implemented in this code.

Download source files - 345 Kb

Download demo - 298 Kb



Introduction

The power of network programming in .NET platform cannot be denied. Socket programming is the core of network programming in Windows and Linux, and today the .NET platform implements it in a powerful way. In this article, we will see the basics of socket programming in C#. To be precise, I have created a command client and a command server, to communicate between a remote

server and up to 200 clients and send the specified commands to them. As a sample application, I have created a chat client application that uses this command client/server to implement chat functions. Before I start explaining my application, let me give you a small introduction on network programming and sockets taken from the book 'C# network programming', written by Richard Blum.

Sockets

In socket-based network programming, you don't directly access the network interface device to send and receive packets. Instead, an intermediary connector is created to handle the programming interface to the network. Assume that a socket is a connector that connects your application to a network interface of your computer. For sending and receiving data to and from the network you should call the socket's methods.

Socket programming in C#

The '`System.Net.Sockets`' namespace contains the classes that provide the actual .NET interface to the low-level Winsock APIs. In network programming, apart from which programming language to use there are some common concepts like the IP address and port. IP address is a unique identifier of a computer on a network and port is like a gate through which applications communicate with each other. In brief, when we want to communicate with a remote computer or a device over the network, we should know its IP address. Then, we must open a gate (Port) to that IP and then send and receive the required data.

IP addresses in C#

One of the biggest advantages you will notice in the .NET network library is the way IP address/port pairs are handled. It is a fairly straightforward process that presents a welcome improvement over the old, confusing UNIX way. .NET defines two classes in the `System.Net` namespace to handle various types of IP address information:

- `IPAddress`
- `IPEndPoint`

IPAddress

An `IPAddress` object is used to represent a single IP address. This value is then used in various socket methods to represent the IP address. The default constructor for `IPAddress` is as follows:

Hide Copy Code

```
public IPAddress(long address)
```

The default constructor takes a `long` value and converts it to an `IPAddress` value. In practice, the default is almost never used. Instead, several methods in the `IPAddress` class can be used to create and manipulate IP addresses. The `Parse()` method is often used to create `IPAddress` instances:

Hide Copy Code

```
IPAddress newaddress = IPAddress.Parse("192.168.1.1");
```

IPEndPoint

The .NET Framework uses the `IPEndPoint` object to represent a specific IP address/port combination. An `IPEndPoint` object is used when binding sockets to local addresses, or when connecting sockets to remote addresses.

Connection-oriented and connectionless sockets

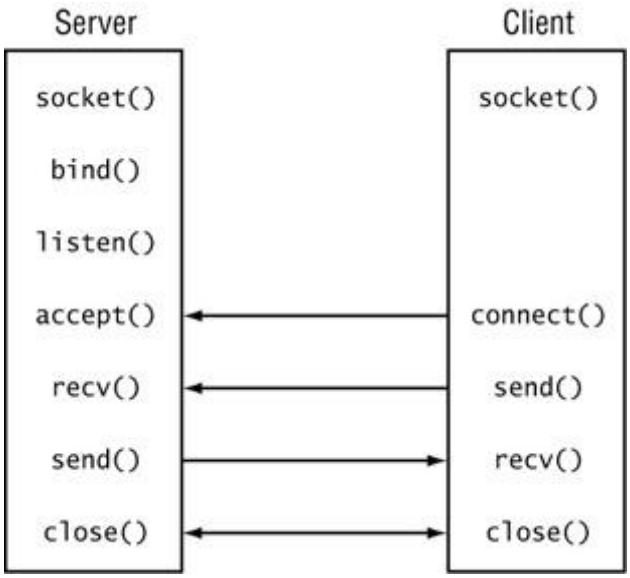
The world of IP connectivity revolves around two types of communication: connection-oriented and connectionless. In a connection-oriented socket, the TCP protocol is used to establish a session (connection) between two IP address endpoints. There is a fair amount of overhead involved with establishing the connection, but once it is established, the data can be reliably transferred

between the devices.

Connectionless sockets use the UDP protocol. Because of that no connection information is required to be sent between the network devices and it is often difficult to determine which device is acting as a "server", and which is acting as a "client". We will focus on the first type of socket programming in this article.

Using connection-oriented sockets

In the .NET Framework, you can create connection-oriented communications with remote hosts across a network. To create a connection-oriented socket, separate sequences of functions must be used for server programs and client programs:



Server

You have four tasks to perform before a server can transfer data with a client connection:

1. Create a socket.
2. Bind the socket to a local `IPEndPoint`.
3. Place the socket in listen mode.
4. Accept an incoming connection on the socket.

Creating the server

The first step to constructing a TCP server is to create an instance of the `Socket` object. The other three functions necessary for successful server operations are then accomplished by using the methods of `Socket` object. The following C# code snippet includes these steps:

Hide Copy Code

```
IPEndPoint localEndPoint = new IPEndPoint(IPAddress.Any, 8000);
Socket newsock = Socket(AddressFamily.InterNetwork,
                  SocketType.Stream, ProtocolType.Tcp);
newsock.Bind(localEndPoint);
newsock.Listen(10);
Socket client = newsock.Accept();
```

The `Socket` object created by the `Accept()` method can now be used to transmit data in either direction between the server and the remote client.

Client

Now that you have a working TCP server, you can create a simple TCP client program to interact with it. There are only two steps required to connect a client program to a TCP server:

1. Create a socket.
2. Connect the socket to the remote server address.

Creating the client

As it was for the server program, the first step for creating the client program is to create a `Socket` object. The `Socket` object is used by the socket `Connect()` method to connect the socket to a remote host:

Hide Copy Code

```
IPEndPoint ipep =
    new IPEndPoint(Ipaddress.Parse("127.0.0.1"), 8000);
Socket server = new Socket(AddressFamily.InterNetwork,
                  SocketType.Stream, ProtocolType.Tcp);
server.Connect(ipep);
```

This example attempts to connect the socket to the server located at address 127.0.0.1.This is the IP address of the local host (current computer) and is a loopback IP for testing a network application without a network. Of course, you can also use hostnames along with the `Dns.Resolve()` method in a real network. (Dns is in `System.Net` namespace). Once the remote server TCP program accepts the connection request, the client program is ready to transmit data with the server using the standard `Send()` and `Receive()` methods.

Blocking problem of network applications

Sockets are in blocking mode by default. In this mode they will wait forever to complete their functions, holding up other functions within the application program until they are complete. Many programs can work quite competently in this mode, but for applications that work in the Windows programming environment, this can be a problem. There are some ways to solve this problem. The first thing that comes to a programmer's mind is multi threading. I chose this solution in my application too. This is a simple way when compared to asynchronous network programming or the old 'Non-Blocking sockets' way.

Our command client/server

After a brief introduction on network programming in C#, I should give you more details about our command client/server application here. Of course, I can't write a book on network programming in this little article. This is only an introduction to network programming. You can find many samples and tutorials on MSDN and CodeProject explaining this concept in detail.

About the command server

The server application is a console program. After starting, it will bind to the '127.0.0.1' local IP and wait on the port 8000 by default for clients. You can pass the IP and the port of the server as the first and second command line parameters when starting the server, if you have a real network. For example: *c:\> ConsoleServer 192.198.0.100 8960*.

I used `BackgroundWorker` to implement multithreading in time consuming functions in the server and client. One of these actions includes the acceptance part of the server:

Hide Copy Code

```
bwListener = new BackgroundWorker();
bwListener.DoWork += new DoWorkEventHandler(StartToListen);
bwListener.RunWorkerAsync();

private void StartToListen(object sender , DoWorkEventArgs e)
{
   this.listenerSocket = new Socket(AddressFamily.InterNetwork,
                           SocketType.Stream, ProtocolType.Tcp);
   this.listenerSocket.Bind(
               new IPEndPoint(this.serverIP , this.serverPort));
   this.listenerSocket.Listen(200);
   while ( true )
      this.CreateNewClientManager(this.listenerSocket.Accept());
}
```

I have a class named `ClientManager`. When the server is connected to a remote client it passes the communication socket to this class and adds this new `ClientManager` object to a list of current connected remote clients. Each `ClientManager` object in the list is responsible for communicating with its remote client. The `ClientManager` object announces the server with various events defined in this class when an action takes place between the remote client and the server. These events are:

Hide Copy Code

```
public event CommandReceivedEventHandler CommandReceived;
```

Occurs when a command is received from a remote client.

Hide Copy Code

```
public event CommandSentEventHandler CommandSent;
```

Occurs when a command had been sent to the remote client successfully.

Hide Copy Code

```
public event CommandSendingFailedEventHandler CommandFailed;
```

Occurs when a command sending action fails. This is may be because of disconnection or sending exception.

Hide Copy Code

```
public event DisconnectedEventHandler Disconnected;
```

Occurs when a client is disconnected from this server.

Sending and receiving data

Since we have a command client/server application we should have a command object to send and receive data. This is implemented in a '`Command`' class. This class is the same in client and server. When the server wants to send a command to the client it builds a `Command` object and then sends it to the client and vice versa.

The `command` class is good for the user of this code. But in the network, we can't send and receive an object or a type. Everything should be converted to `byte` array. So, we should convert this object to a `byte` array part by part and send or receive it over the network in real `Send` and `Receive` functions inside our code. The following code shows the `send` command method. '`cmd`' is the command that we want to send to the remote client:

Hide Shrink Copy Code

```
//Type
byte [] buffer = new byte [4];
buffer = BitConverter.GetBytes((int)cmd.CommandType);
this.networkStream.Write(buffer , 0 , 4);
this.networkStream.Flush();

//Sender IP
byte [] senderIPBuffer =
     Encoding.ASCII.GetBytes(cmd.SenderIP.ToString());
buffer = new byte [4];
buffer = BitConverter.GetBytes(senderIPBuffer.Length);
this.networkStream.Write(buffer , 0 , 4);
this.networkStream.Flush();
this.networkStream.Write(senderIPBuffer, 0,
                                 senderIPBuffer.Length);
this.networkStream.Flush();

//Sender Name
byte [] senderNameBuffer =
   Encoding.Unicode.GetBytes(cmd.SenderName.ToString());
buffer = new byte [4];
buffer = BitConverter.GetBytes(senderNameBuffer.Length);
this.networkStream.Write(buffer , 0 , 4);
this.networkStream.Flush();
this.networkStream.Write(senderNameBuffer, 0,
                                 senderNameBuffer.Length);
this.networkStream.Flush();
```

```
//Target
byte [] ipBuffer =
  Encoding.ASCII.GetBytes(cmd.Target.ToString());
buffer = new byte [4];
buffer = BitConverter.GetBytes(ipBuffer.Length);
this.networkStream.Write(buffer , 0 , 4);
this.networkStream.Flush();
this.networkStream.Write(ipBuffer , 0 , ipBuffer.Length);
this.networkStream.Flush();

//Meta Data.
if ( cmd.MetaData == null || cmd.MetaData == "" )
    cmd.MetaData = "\n";

byte [] metaBuffer =
          Encoding.Unicode.GetBytes(cmd.MetaData);
buffer = new byte [4];
buffer = BitConverter.GetBytes(metaBuffer.Length);
this.networkStream.Write(buffer , 0 , 4);
this.networkStream.Flush();
this.networkStream.Write(metaBuffer, 0, metaBuffer.Length);
this.networkStream.Flush();
```

The `send` and `receive` are bidirectional operations. For example, when we send 4 bytes to the client, the client should read the 4 bytes. We should repeat this operation until all the sent data is read. See the receive code of the client here:

Hide Shrink ▲ Copy Code

```
while ( this.clientSocket.Connected )
{
    //Read the command's Type.
    byte [] buffer = new byte [4];
    int readBytes = this.networkStream.Read(buffer , 0 , 4);
    if ( readBytes == 0 )
      break;
    CommandType cmdType =
         (CommandType)( BitConverter.ToInt32(buffer , 0) );

    //Read the command's sender ip size.
    buffer = new byte [4];
    readBytes = this.networkStream.Read(buffer , 0 , 4);
    if ( readBytes == 0 )
    break;
    int senderIPSize = BitConverter.ToInt32(buffer , 0);

    //Read the command's sender ip.
    buffer = new byte [senderIPSize];
    readBytes =
        this.networkStream.Read(buffer , 0 , senderIPSize);
    if ( readBytes == 0 )
      break;
    IPAddress senderIP = IPAddress.Parse(
             System.Text.Encoding.ASCII.GetString(buffer));

    //Read the command's sender name size.
    buffer = new byte [4];
    readBytes = this.networkStream.Read(buffer , 0 , 4);
    if ( readBytes == 0 )
      break;
    int senderNameSize = BitConverter.ToInt32(buffer , 0);

    //Read the command's sender name.
    buffer = new byte [senderNameSize];
    readBytes = this.networkStream.Read(buffer, 0, senderNameSize);
    if ( readBytes == 0 )
      break;
    string senderName =
        System.Text.Encoding.Unicode.GetString(buffer);

    //Read the command's target size.
    string cmdTarget = "";
    buffer = new byte [4];
    readBytes = this.networkStream.Read(buffer , 0 , 4);
    if ( readBytes == 0 )
```

```
        break;
    int ipSize = BitConverter.ToInt32(buffer , 0);

    //Read the command's target.
    buffer = new byte [ipSize];
    readBytes = this.networkStream.Read(buffer , 0 , ipSize);
    if ( readBytes == 0 )
      break;
    cmdTarget = System.Text.Encoding.ASCII.GetString(buffer);

    //Read the command's MetaData size.
    string cmdMetaData = "";
    buffer = new byte [4];
    readBytes = this.networkStream.Read(buffer , 0 , 4);
    if ( readBytes == 0 )
      break;
    int metaDataSize = BitConverter.ToInt32(buffer , 0);

    //Read the command's Meta data.
    buffer = new byte [metaDataSize];
    readBytes = this.networkStream.Read(buffer , 0 , metaDataSize);
    if ( readBytes == 0 )
         break;
    cmdMetaData = System.Text.Encoding.Unicode.GetString(buffer);

    Command cmd = new Command(cmdType,
                        IPAddress.Parse(cmdTarget), cmdMetaData);
    cmd.SenderIP = senderIP;
    cmd.SenderName = senderName;
    this.OnCommandReceived(new CommandEventArgs(cmd));
    }
    this.OnServerDisconnected(new ServerEventArgs(this.clientSocket));
    this.Disconnect();
}
```

About the command client

The command client is very similar to the server. Everything is in the '`CommandClient`' class. Since our application is an event driven program this class also has some events to announce the user of the occurred actions. Here is a brief definition of these events:

Hide Copy Code

```
public event CommandReceivedEventHandler CommandReceived;
```

Occurs when a command is received from a remote client.

Hide Copy Code

```
public event CommandSentEventHandler CommandSent;
```

Occurs when a command has been sent to the remote server successfully.

Hide Copy Code

```
public event CommandSendingFailedEventHandler CommandFailed;
```

Occurs when a command sending action fails. This is because of disconnection or sending exception.

Hide Copy Code

```
public event ServerDisconnectedEventHandler ServerDisconnected;
```

Occurs when the client is disconnected.

Hide Copy Code

```
public event DisconnectedEventHandler DisconnectedFromServer;
```

Occurs when this client is disconnected from the remote server.

Hide Copy Code

```
public event ConnectingSuccessedEventHandler ConnectingSuccessed;
```

Occurs when this client is connected to the remote server successfully.

Hide Copy Code

```
public event ConnectingFailedEventHandler ConnectingFailed;
```

Occurs when this client fails on connecting to the server.

Hide Copy Code

```
public event NetworkDeadEventHandler NetworkDead;
```

Occurs when the network fails.

Hide Copy Code

```
public event NetworkAlivedEventHandler NetworkAlived;
```

Occurs when the network starts to work.

Conclusion

In this application, you can find the following concepts of .NET programming:

- Socket programming, server side and client side.
- Working with resources at runtime.
- Concurrency management in multi threaded environment.
- Calling windows API functions within C# code.
- Creating custom events and eventargs, and throwing events in a UI safe mode.
- Creating custom exceptions and throwing them as and when needed.
- Generating an HTML page at runtime dynamically.

And many other .NET programming concepts that I couldn't explain in detail here. The code is fully XML commented and very clear to understand. Please contact me if there is any ambiguous point or you need any help on my code.

License

Share

About the Author

ALLEXY

Software Developer (Senior)

Australia

MCPD
MCTS: ASP.NET Applications
MCTS: ADO.NET Applications
MCTS: Windows Forms Applications
MCTS: Windows Communication Foundation Applications
MCPD: Enterprise Application Developer

Comments and Discussions

c

A project with an Output type of class Library can not be started directly
in order to debug this project, add an executable project to this solution which reference the library project. set the executable project as the startup project

Sign In · View Thread · Permalink 1.00/5 (1 vote)

main file

which file I should first run on the server side and client side?

Sign In · View Thread · Permalink 1.00/5 (1 vote)

IOException was unhandled by user code

When I run run the Server side program and connect a Client, I am getting this (Unable to read data from the transport connection: An existing connection was forcibly closed by the remote host.) exception on this line:
readBytes = this.networkStream.Read(buffer, 0, 4);
Please help me.

Sign In · View Thread · Permalink

How can we hide command window?

Thank you for your clean codes. But I couldn't success to hide command window. Can you help me?

Sign In · View Thread · Permalink

Re: How can we hide command window?

Re: How can we hide command window?

My vote of 1

As a Developer and Programmer, I find this chat and its topic very Misleading. I will be the first to give Credits and Say a few good works to other amazing and hardworking developers. But in this case I can see that many weeks of hardwork has been totally waste of time.
About the Chat: Users cannot Join the same room. Which means NO ONE can chat with eachother using this Chat Application. And No one can send PM to eachtoher. Users can only Chat with themselfs ALONE and SEND pm to themselfs only. I mean really? So whats the Poibnt of this Chat when you cant chat with others than yourself? Plus youc ant even join with multiple Usersnames in the Chat... Thats why I give this

Sign In · View Thread · Permalink

Not a Real Chat Application, Users cant Chat With Eachother, And cant join the same room (Misleading)

When you think about the word CHAT then you think about Chatting with other People, and Joining the Same room. But in this Chat, Thats not possible.

I understand that the developer went this far to develope an app. But why not finish it?

- You cant Chat with others, but yourself only.
- You cant send Private Messages to others, but yourself only.
- Other Users cannot join your room or other rooms.
- When you join the room, then thats about it. There are no chance that you will see 2 (Two) Usernames in the same room.

- In order to Chat with other users, you will have to complete change the codings process and recode the whole project

I trully hope someone else could take over and make this to a Real Chat Application so Users can Join the same Room or Chat in a Private Message.

modified 5 days ago.

Sign In · View Thread · Permalink

Re: Not a Real Chat Application, Users cant Chat With Eachother, And cant join the same room (Misleading)

Re: Not a Real Chat Application, Users cant Chat With Eachother, And cant join the same room (Misleading)

Request for make Chat application over the internet

Hello ALLEXY,
Please can you modify you chat application working over the internet.
i will very thankful to you. I have already Static IP but when i replace with 127.0.0.1 its not working, even i fixed port forwarding.

Please help me about this issue .

Regard 

Sign In · View Thread · Permalink

Code

Hi i have been looking over you code and would like to convert it to VB.net to be part of a project i am working on, have you written this in vb.net already? if not would you be able to help me convert it to vb.net ?

Many Regards

Pete

Sign In · View Thread · Permalink

Re: Code

Hey IP problem

Hi, everything works absolutely fine, real well good job that you have done. However, it is constantly wanting to use the 127.... address which, obviously isn't my IP. I have tried changing it in the chat client (both places), and it just won't, then, connect. Any idea what I'm doing wrong?

cheers

Sign In · View Thread · Permalink 1.00/5 (1 vote)

Re: Hey IP problem

Great !!!

Thanks for your tutorial! I understood clearly your program's idea 

Sign In · View Thread · Permalink

Help me plz

I actually i downloaded ur app, but i am unable to use it. It doesn't start chat after login, it sends message to me.

And i also have to know, will it work in entire world.

Plz help me.

Sign In · View Thread · Permalink

it works for only one user

I opened two client window , I noticed that users was able to chat with itself but not with each-other , you must code to help users to chat with each - other

Sign In · View Thread · Permalink 1.00/5 (1 vote)

Re: it works for only one user

Re: it works for only one user

Re: it works for only one user

Shared window

hi allexy dere is public message window after login, which i think, should share the messages among online members bt its not like dt instead m seeing only dt wtever i m posting eg login by two accounts (A & B) by creating two new instances of chat client . What A send is seeing on A public window bt not on B n vice versa...Plz help me out

Sign In · View Thread · Permalink

Unable to read data from the transport connection: A blocking operation was interrupted by a call to WSACancelBlockingCall.

Hi

Thanks for your code, i just want to inform you btw that when i tried to close the client or the server the client is having an error "Unable to read data from the transport connection: A blocking operation was interrupted by a call to WSACancelBlockingCall."
The error occurs at

Hide Copy Code

```
private void StartReceive(object sender, DoWorkEventArgs e)
```

of the

Hide Shrink ▲ Copy Code

```

```

CMDClient class, i think the backgroundworker is still running even if the code already cancelled the bwReceiver in the

Hide Copy Code

```
public bool Disconnect()
```

Method.

Hide Copy Code

```
this.bwReceiver.CancelAsync();
```

IMO

modified 6-Nov-13 3:59am.

Sign In · View Thread · Permalink

Server Is Not Acessible

As I click Login, it gives the error message, the hand of the clock, "SERVER IS NOT acessible" And then I have to go there in the folder CommandServer \ ConsoleServer \ bin \ Debug and click on the file to get CommandServer log, there is another way through code it directly log on the server?

Sign In · View Thread · Permalink

Help Me

Hide Copy Code

```
I already have a solution and wanted to know how to put this project in my solution, because I
put up, but does not connect to the server, I have to debug it in the folder and then click the
file to connect. How do I get it to be automatic?

I'm starting now OK?!

Thank you.
```

Sign In · View Thread · Permalink

My vote of 5

Excellent

Sign In · View Thread · Permalink

My vote of 3

AddressFamily is ambigious

Sign In · View Thread · Permalink

My vote of 5

Awesome Post...Thanks

Sign In · View Thread · Permalink

Does not run

This solution does not run in Visual Studio 2012.
CommandClient throws an error saying 'A project with an Output Type of Class Library can not be started directly.

Sign In · View Thread · Permalink 5.00/5 (1 vote)

Re: Does not run

Re: Does not run

My vote of 5

Excellent....Wonderful....Awesome...!!!!!

Sign In · View Thread · Permalink

execute procedure.

Can you tell me the procedure of how to execute the source file i.e. i download the source file but don"t no how to run . Please tell me the procedure of execute.

Thank you in advance.

my email id is ankitjainkar@gmail.com

Sign In · View Thread · Permalink

Re: execute procedure.

Re: execute procedure.

nice code but i have problems

The code is neat and well explained, there's just one problem: it does not work as intended.
1. the client form does not register other clients in the list to the left in the client window
2. when you start multiple clients and try to private chat with yourself(since there is no one else in the list as mentioned above) the message goes through to all clients (including yourself) by opening new private chat windows for each client.
Basically chaos everywhere

I tried debugging and building it myself, and running the demo available for download. Same result both times(the one explained above)

Can you please tell me if I am doing something wrong or if the code is missing something or whatever i just need to make it work somehow. It would mean a great deal to me.
Thanks in advance.

Sign In · View Thread · Permalink

Nice code but i got a problem

The code is neat and well explained, there's just one problem: it does not work as intended.
1. the client form does not register other clients in the list to the left in the client window
2. when you start multiple clients and try to private chat with yourself(since there is no one else in the list as mentioned above) the message goes through to all clients (including yourself) by opening new private chat windows for each client.
Basically chaos everywhere

I tried debugging and building it myself, and running the demo available for download. Same result both times(the one explained above)

Can you please tell me if I am doing something wrong or if the code is missing something or whatever i just need to make it work somehow. It would mean a great deal to me.
Thanks in advance.

Sign In · View Thread · Permalink

Excellent!!

Excellent

Sign In · View Thread · Permalink 5.00/5 (1 vote)

more explanation required

The article is really very good, but it was a bit confusing for me to understand ClientManager and Command class. It's bit tough for beginners and some intermediate as well. But still it is very good. Specially your events are addressed very well.
Thanks.

TheRaaaZ

Sign In · View Thread · Permalink 5.00/5 (2 votes)

My vote of 5

آقا خیلی مخلسیم حرسم داشت در میومد ینفر ایرانی تو این سایت نبود واقعن دمت گرم پرچم بردی بالا ایول

Sign In · View Thread · Permalink 1.00/5 (3 votes)

Re: My vote of 5

My vote of 5

It's exactly my answer!

Sign In · View Thread · Permalink

My vote of 5

Great job!

Sign In · View Thread · Permalink

Fantastic example!

After looking at half the posts here, I felt I had to post something as well.
First and foremost, I believe this is a great example! It has been well structured, the reasoning has been explained, and this gives you a basic framework to send every sort of command you can think of (well, most anyway).
After making lots of little C# programs, I really wanted to see about making some basic network driven stuff, and this has been just that. Thank you for that!

As for all the idiots who can't distinguish between an example/learning article and full-proof, enterprise grade, cloud-scalable source code, and still complain about it... They should simply keep on programming Hello World! applications, and read some more books until they actually get what they're talking about.

Anyway, kudo's and many thanks!

Sign In · View Thread · Permalink

How I can debug source code?

Dear author,

I can't debug your source code because one reference (CommandClient) is not on there?
how i can fix it?
Please anyone who can help, please answer my question.

Sign In · View Thread · Permalink 2.67/5 (3 votes)

Re: How I can debug source code?

Got server to work in VS 2012!

I made a new solution with the two projects in it. CommandServer needs to be Output Type 'Class Library', and for ConsoleServer I had to add a reference for CommandServer in addition to the using directive...

Sign In · View Thread · Permalink 2.00/5 (4 votes)

My vote of 5

awesome~

Sign In · View Thread · Permalink 2.33/5 (3 votes)

My vote of 5

this is very very good

Sign In · View Thread · Permalink

help

Hide Copy Code

```
while (true)
    {
        try
        {
            Console.WriteLine("循环低" + i.ToString() + "次");
            this.CreateNewClientManager(this.listenerSocket.Accept());
        }
        catch(Exception a)
        {
            Console.WriteLine(a.Message.ToString());
        }
    }
```

why it don't loop until a client started?

Sign In · View Thread · Permalink