Global Text Chat Room Application using C#.NET Remoting Technology

SumanBiswas, 26 Oct 2012 LGPL3 Rate:

4.35 (20 votes)

Chat Room using C#.NET Remoting

Download source - 70.56 KB

Visit my personal technical blog on Socket Programming in C#

Introduction

This is a chat room application. It can run on LAN or Internet. When user logs in to that server, then he/she will be automatically logged in a default chat room. Every one can chat within that room and their messages will be broadcasted in that room.

User Manual

To run the application, first you have to run the server application. Then you will get one Window with two buttons 'Start' and 'Stop' as screenshot. Your have to run the server by pressing the Start button.

Now run the client application. Here you will first get a Window which will ask your name and server address, port, etc. If you try with the same machine, then the given address is ok. But if it is a different address, then you need to update '`localhost`' with a particular IP address or host name. But do not change the port number and remaining things and press enter or click on join button. See the screenshot.

Now the chat room window will appear. Here there are three sections. The left and large one is the chat history, the right list is for online users, the bottom area for entering your text and send button for sending message to the server.

Remoting Technology - Basic and Simple Architecture

The basic and simple architecture of .NET Remoting technology has three parts. These are:

1. **Base Remoting class**: This is like a bridge to communicate between Client and Server. It exists in a DLL file which shares Server and Client program.
2. **Server class**: It is a server to serve Client requests. Every client connects to Server to communicate with each other. This program holds a Remoting class's DLL.
3. **Client class**: This is the client part of Remoting architecture. This also holds a copy of Remoting base DLL. It connects to Server and via server communicates to other client.

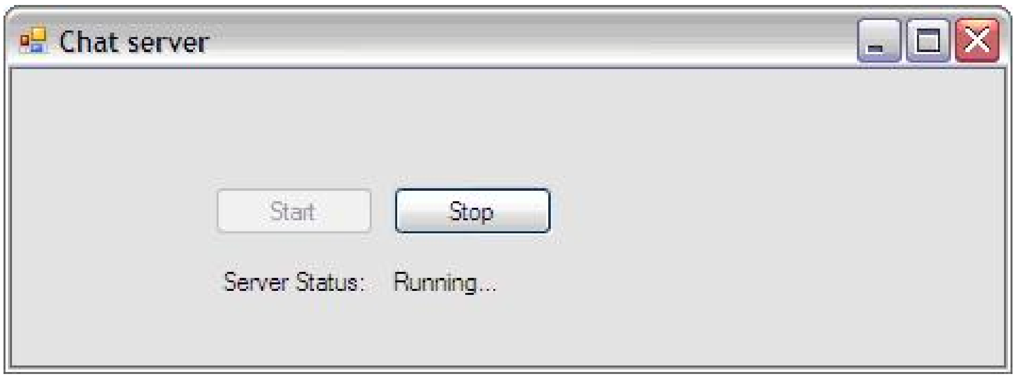
This is very simple idea of Remoting architecture. If you want to learn about this technology, then you may read from the MSDN site.

About the Application

I will describe a Global Text Chat Room application using this technology. This is very easy to develop and interesting also.

As Remoting architecture here has a base class (class library) and after compiling, produces a DLL file with name '*RemoteBase.dll*'. This DLL has about six methods like:

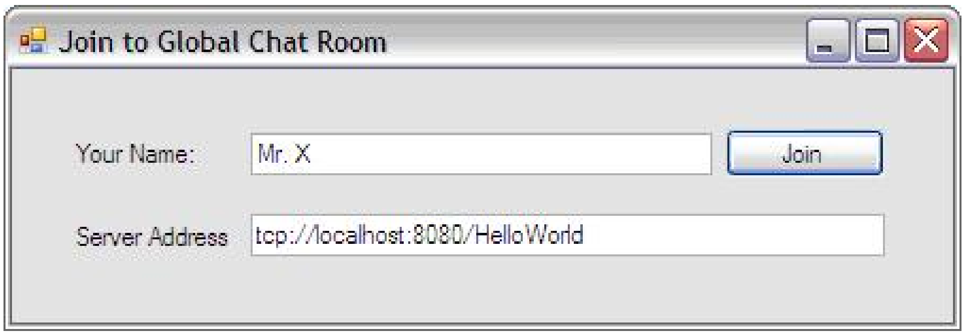
- `JoinToChatRoom`
- `LeaveChatRoom`
- `SendMsgToSvr` (Send Message To Server)
- `GetMsgFromSvr` (Get Message From Server), etc.



Server Status: Running...

The next one is the Server, this is a Windows Form (WinForms) application. This application uses '*RemoteBase.dll*' as its reference file for library. Server registers a TCP channel with a port number. You may choose any port number from 1025 to 65k. And it registers for well known type of `RemoteBase` and mode type is `Singleton`. (Remember it should not work for `Singlecall` type, details and differences will be found on MSDN).

When you run the server, you will see a window as the attached screen shot and need to press button 'Start' to start the server and check server status as 'Running'. To stop the server, you need to press on 'Stop' button.

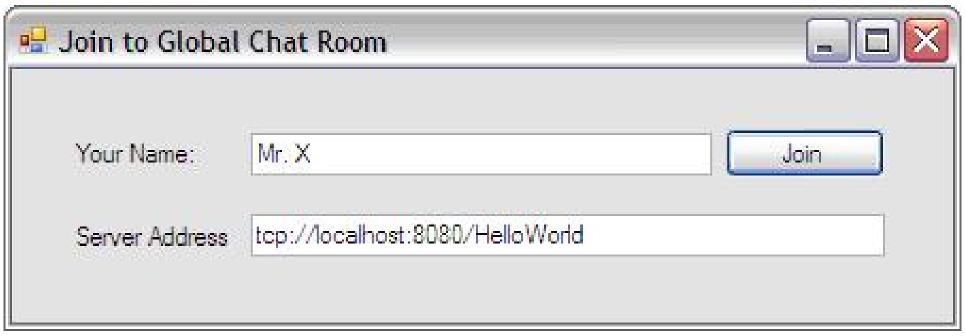


Your Name: Mr. X

Server Address tcp://localhost:8080/HelloWorld

The last one is the Client part, it is also a Windows form (`WinForm`) application with two forms. Like the server, client also takes reference of '*RemoteBase.dll*' for library. When you run this client application, one popup window will come and ask for your name which will be used in the chat room to represent you. Then press on Join button. After that, the chat room window will open.

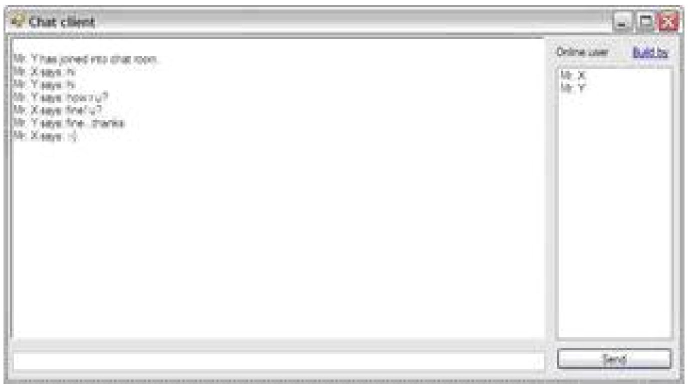
There also is a server address like '*tcp://localhost:8080/HelloWorld*' here 'localhost' is the server address and 8080 is the port number. Server address needs to tell where your server is running. I am using server and client in the same machine so server address is 'localhost', you may give any IP address here. Port number can also be changed but server opening port number and client requesting port number should be the same. You cannot change the remaining things in address, otherwise this chat application will not work.


Join to Global Chat Room

Your Name: Mr. X

Server Address tcp://localhost:8080/HelloWorld

Chat room window has four sections, the largest one to see all chat messages and below that to type chat message, and send button to send message to server. Just above is the list to display all online users.


Chat client

Mr. Y has joined into chat room.
Mr. X says: hi
Mr. Y says: hi
Mr. Y says: how r u?
Mr. X says: fine/ u?
Mr. Y says: fine.. thanks
Mr. X says: :)

Online user

Mr. X
Mr. Y

When you put your name, then client application creates a remote base class's object and connects to server by registering TCP channel. Then it connects to Chat Room and seeks the latest message number. After that, the main Chat Room window opens. From that window, it seeks the latest available message in the server by a timer. To get message from server, it invokes '`GetMsgFromSvr()`', and gets available online user through '`GetOnlineUser()`, and user message sends from client application to server by invoking '`SendMsgToSvr()`'. For better understanding, you may go through the code.

How the Code is Working

Start the Server

When users are trying start Server by clicking Start button, the following code executes:

Hide Copy Code

```
private void btnStart_Click(object sender, EventArgs e)
{
if (channel == null)
{
channel = new TcpChannel(8080);
ChannelServices.RegisterChannel(channel, false);
RemotingConfiguration.RegisterWellKnownServiceType
    (typeof(SampleObject), "ChatRoom", WellKnownObjectMode.Singleton);
lblStatus.Text = "Running...";
btnStart.Enabled = false;
btnStop.Enabled = true;
}
}
```

Here a `TcpChannel` opens with port number 8080 and register it as `WellKnownServiceType`. '`ChatRoom`' is the '`ObjectUri`' it will require to connect to server from client.

On the other hand, when user presses 'Stop' server, then unregister the channel and stop the server. The code is shown below:

Hide Copy Code

```
private void btnStop_Click(object sender, EventArgs e)
{
if (channel != null)
{
ChannelServices.UnregisterChannel(channel);
channel = null;
lblStatus.Text = "Stopped.";
btnStart.Enabled = true;
btnStop.Enabled = false;
}
}
```

Join ChatRoom

Next comes the client, how client connects to server and user logs in to the chat room. To join the chat room, the below code is executed:

Hide Copy Code

```
private void JoinToChatRoom()
{
if (chan == null && txtName.Text.Trim().Length != 0)
{
chan = new TcpChannel();
ChannelServices.RegisterChannel(chan,false);

// Create an instance of the remote object
objChatWin = new frmChatWin();
objChatWin.remoteObj = (SampleObject)Activator.GetObject
            (typeof(RemoteBase.SampleObject), txtServerAdd.Text);

if (!objChatWin.remoteObj.JoinToChatRoom(txtName.Text))
{
MessageBox.Show(txtName.Text+ " already joined, please try with different name");
ChannelServices.UnregisterChannel(chan);
chan = null;
objChatWin.Dispose();
return;
}
objChatWin.key = objChatWin.remoteObj.CurrentKeyNo();

objChatWin.yourName= txtName.Text;

this.Hide();
objChatWin.Show();
}
}
```

Here from the user, the client application takes a name and checks with the server if the name is available or not. If the name is available, then user gets the '`CurrentKeyNo`' of server, it is the number of last chat message (how the key is generated, described later) and open the ChatRoom window.

If user name is already taken by another user, then the application asks the user for a different name.

On the server side to join a user in chat server "`JoinToChatRoom()`" method is invoked. Let's see what happens within the method:

Hide Copy Code

```
public bool JoinToChatRoom(string name)
{
if (alOnlineUser.IndexOf(name) > -1)
return false;
else
{
alOnlineUser.Add(name);
SendMsgToSvr(name + " has joined into chat room.");
return true;
}
}
```

Here the user can successfully log in to the server, then his name is added in a user collection, '`alOnlineUser`' is an `ArrayList` type object.

Send Message to Server

When user types some message and presses on the 'Send' button or just presses 'Enter' button, then the client application tries to send a message to the server. To do it, the client calls the below method:

Hide Copy Code

```
private void SendMessage()
{
if (remoteObj != null && txtChatHere.Text.Trim().Length>0)
{
remoteObj.SendMsgToSvr(yourName + " says: " + txtChatHere.Text);
txtChatHere.Text = "";
}
}
```

Here client application invokes the "`SendMsgToSvr()`" method of the server. Let's see what the method is doing:

Hide Copy Code

```
public void SendMsgToSvr(string chatMsgFromUsr)
{
hTChatMsg.Add(++key, chatMsgFromUsr);
}
```

Wow! This is very small code. Actually this is adding the users' message to another collection. I have used for this collection of `HashTable` type, you may use any other collection type to store `string` data.

See here there is one counter 'key' which is incrementing by one. This is the counter which is maintaining the chat message number. It will help us to get the chat message from server.

Receive Message from Server Ok friends, next we look at how data is got from the server.

In chat room a timer always fires which tries to get a message from the server and current available user in server. Here the below code plays in the Client side:

Hide Copy Code

```
private void timer1_Tick(object sender, EventArgs e)
{
if (remoteObj != null)
{
string tempStr = remoteObj.GetMsgFromSvr(key);
if (tempStr.Trim().Length > 0)
{
key++;
txtAllChat.Text = txtAllChat.Text + "\n" + tempStr;
}

ArrayList onlineUser = remoteObj.GetOnlineUser();
lstOnlineUser.DataSource = onlineUser;
skipCounter = 0;

if (onlineUser.Count < 2)
{
txtChatHere.Text = "Please wait until atleast two user join in Chat Room.";
txtChatHere.Enabled = false;
}
else if(txtChatHere.Text == "Please wait untill atleast two user join in Chat Room."
                && txtChatHere.Enabled == false)
{
txtChatHere.Text = "";
txtChatHere.Enabled = true;
}
}
}
```

Here client invokes "`GetMsgFromSvr()`" method to get a message, with parameter key. The code of the method is:

Hide Copy Code

```
public string GetMsgFromSvr(int lastKey)
{
if (key > lastKey)
return hTChatMsg[lastKey + 1].ToString();
else
return "";
}
```

The server just takes the key as last-key of user's message and uses it in Chat message collection to fetch the next chat message, after that the message returns to the client.

To online user client application invokes "`GetOnlineUser()`" method, let's see what happens in the server side within the method.

Hide Copy Code

```
public ArrayList GetOnlineUser()
{
return alOnlineUser;
}
```

So this method returns the user collection object which was created in "`JoinToChatRoom()`" method.

Leave the Chat Room

When user closes the chat room, then client application requests the server to remove his/her name from server's online user list. On the client side, the below method invokes:

Hide Copy Code

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
if (remoteObj != null)
{
remoteObj.LeaveChatRoom(yourName);
txtChatHere.Text = "";
}
Application.Exit();
}
```

On the server side, the "`LeaveChatRoom()`" method is invoked. The code of that method is:

Hide Copy Code

```
public void LeaveChatRoom(string name)
{
alOnlineUser.Remove(name);
SendMsgToSvr(name + " has left the chat room.");
}
```

Now the server has deleted your name from the online user list.

You can visit my website to learn more

License

Share

EMAIL TWITTER

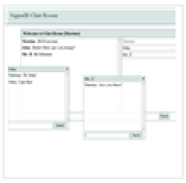
About the Author

SumanBiswas

Web Developer
India

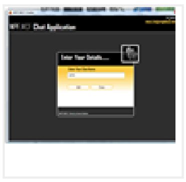
No Biography provided

You may also be interested in...



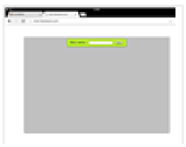
Asp.Net SignalR Chat Room

Securing Your Worklight Applications with IBM Worklight Application Scanning



WCF / WPF Chat Application

Why NoSQL?



MVC 4 Chat room

Beyond RDBMS: A Guide To NoSQL Databases

Comments and Discussions

Add a Comment or Question Search Comments Go

First Prev Next

Plz help new
shahulmechery 24-Jan-15 2:42

Perfect. new
Member 11306862 12-Dec-14 16:55

My vote of 5 new
harishankar.smile 13-Jan-13 9:12

My vote of 5 new
Aslam Iqbal 7-Aug-12 1:36

Hi, quick question on integrating this into another application new
Stutcov 11-Jul-12 14:15

How can I improve this codes to add private chat from any person who add in listbox ? new
Rahi_e 31-Mar-10 3:35

Re: How can I improve this codes to add private chat from any person who add in listbox ? new
Dave Sheets 30-Jun-10 15:40

Re: How can I improve this codes to add private chat from any person who add in listbox ? new
Gary Noter 16-Aug-10 16:56

Using across the internet new
zidane_143 24-Feb-10 13:33

Why not Windows Communication Foundation new
GurliGebis 5-Dec-09 10:50

Re: Why not Windows Communication Foundation new
Ravi Bhavnani 5-Dec-09 12:30

Re: Why not Windows Communication Foundation new
Wb-FreeKill 7-Dec-09 6:51

Why WCF (Re: Why not Windows Communication Foundation) new

SAKryukov 10-Dec-09 21:01

Why WCF (Re: Why not Windows Communication Foundation) [modified] new
SAKryukov 10-Dec-09 21:00

Refresh 1

General News Suggestion Question Bug Answer Joke Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

Permalink | Advertise | Privacy | Terms of Use | Mobile
Web04 | 2.8.150901.1 | Last Updated 26 Oct 2012

Выбрать язык ▼

Layout: fixed | fluid