



Your Company Name

Tel: +44 1234 567 9898

Fax: +44 1234 545 9999

email: info@@company.com

Microsoft Visual Studio

C# Project

Source Code Output

Created using

VScodePrint

Macro Variables Substitution Example

General Date	24/01/2017 6:41:20 PM
Long Date	Tuesday, 24 January 2017
Short Date	24/01/2017
Long Time	6:41:20 PM
Short Time	18:41
Today	24/01/2017 6:41:20 PM
Project Name	ServerApp

Prepared by

Joginder S Nahil

on

24/01/2017 6:41:20 PM

WWW.STARPRINTTOOLS.COM

ClientApp	3
ClientApp	3
Form1.cs	3
ClientApp	3
Form1	3
Form1	3
RtbClientKeyDown	3
SendMessage	3
Program.cs	4
ClientApp	4
Program	4
Main	4
ServerApp	6
Form1.cs	6
ServerApp	6
Form1	6
Form1	6
Server	6
ListenForClients	6
HandleClientComm	6
WriteMessage	7
Echo	7
Program.cs	8
ServerApp	8
Program	8
Main	8

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Net;
8  using System.Net.Sockets;
9  using System.Text;
10 using System.Windows.Forms;
11
12 namespace ClientApp
13 {
14     public partial class Form1 : Form
15     {
16         private string myMessage = "";
17         private TcpClient client = new TcpClient();
18         private IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Parse(
19             "127.0.0.1"), 3000);
20
21         public Form1()
22         {
23             InitializeComponent();
24             client.Connect(serverEndPoint);
25         }
26
27         private void RtbClientKeyDown(object sender, KeyEventArgs e)
28         {
29             if (e.KeyData != Keys.Enter || e.KeyData != Keys.Return)
30             {
31                 myMessage += (char)e.KeyValue;
32             }
33             else
34             {
35                 SendMessage(myMessage);
36                 myMessage = "";
37             }
38         }
39
40         private void SendMessage(string msg)
41         {
42             NetworkStream clientStream = client.GetStream();
43
44             ASCIIEncoding encoder = new ASCIIEncoding();
45             byte[] buffer = encoder.GetBytes(msg);
46
47             clientStream.Write(buffer, 0, buffer.Length);
48             clientStream.Flush();
49
50             // Receive the TcpServer.response.
51
52             // Buffer to store the response bytes.
53             Byte[] data = new Byte[256];
54
55             // String to store the response ASCII representation.
56             String responseData = String.Empty;
57
58             // Read the first batch of the TcpServer response bytes.
59             Int32 bytes = clientStream.Read(data, 0, data.Length);
60             responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes);
61
62             rtbClient.AppendText(Environment.NewLine + "From Server: " +
63                 responseData);
64         }
65     }
66 }
67
68
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Windows.Forms;
5
6  namespace ClientApp
7  {
8      static class Program
9      {
10         /// <summary>
11         /// The main entry point for the application.
12         /// </summary>
13         [STAThread]
14         static void Main()
15         {
16             Application.EnableVisualStyles();
17             Application.SetCompatibleTextRenderingDefault(false);
18             Application.Run(new Form1());
19         }
20     }
21 }
```

- A**
AppendText, 3
Application, 4
ASCII, 3
ASCIIEncoding, 3
- B**
buffer, 3
bytes, 3
- C**
client, 3
ClientApp, 3, 4
clientStream, 3
Collections, 3, 4
ComponentModel, 3
Connect, 3
- D**
data, 3
Data, 3
Drawing, 3
- E**
e, 3
Empty, 3
EnableVisualStyles, 4
encoder, 3
Encoding, 3
Enter, 3
Environment, 3
- F**
Flush, 3
Form, 3
Form1, 3, 4
Forms, 3, 4
- G**
Generic, 3, 4
GetBytes, 3
GetStream, 3
GetString, 3
- I**
InitializeComponent, 3
Int32, 3
IPAddress, 3
IPEndPoint, 3
- K**
KeyData, 3
KeyEventArgs, 3
Keys, 3
KeyValue, 3
- L**
Length, 3
Linq, 3, 4
- M**
Main, 4
msg, 3
myMessage, 3
- N**
Net, 3
NetworkStream, 3
NewLine, 3
- P**
Parse, 3
partial, 3
Program, 4
- R**
Read, 3
responseData, 3
rtbClient, 3
RtbClientKeyDown, 3
Run, 4
- S**
sender, 3
SendMessage, 3
serverEndPoint, 3
SetCompatibleTextRenderingDefault, 4
Sockets, 3
STAThread, 4
System, 3, 4
- T**
TcpClient, 3
Text, 3
- W**
Windows, 3, 4
Write, 3

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Net;
8  using System.Net.Sockets;
9  using System.Text;
10 using System.Threading;
11 using System.Windows.Forms;
12
13 namespace ServerApp
14 {
15     public partial class Form1 : Form
16     {
17         private TcpListener tcpListener;
18         private Thread listenThread;
19         private int connectedClients = 0;
20         private delegate void WriteMessageDelegate(string msg);
21
22         public Form1()
23         {
24             InitializeComponent();
25             Server();
26         }
27
28         private void Server()
29         {
30             this.tcpListener = new TcpListener(IPAddress.Loopback, 3000); // Change to IPAddress.Any for internet wide Communication
31             this.listenThread = new Thread(new ThreadStart(ListenForClients));
32             this.listenThread.Start();
33         }
34
35         private void ListenForClients()
36         {
37             this.tcpListener.Start();
38
39             while (true) // Never ends until the Server is closed.
40             {
41                 //blocks until a client has connected to the server
42                 TcpClient client = this.tcpListener.AcceptTcpClient();
43
44                 //create a thread to handle communication
45                 //with connected client
46                 connectedClients++; // Increment the number of clients that have communicated with us.
47                 lblNumberOfConnections.Text = connectedClients.ToString();
48
49                 Thread clientThread = new Thread(new ParameterizedThreadStart(HandleClientComm));
50                 clientThread.Start(client);
51             }
52         }
53
54         private void HandleClientComm(object client)
55         {
56             TcpClient tcpClient = (TcpClient)client;
57             NetworkStream clientStream = tcpClient.GetStream();
58
59             byte[] message = new byte[4096];
60             int bytesRead;
61
62             while (true)
63             {
64                 bytesRead = 0;
65
66                 try
67                 {
68                     //blocks until a client sends a message

```

1 2 3 4

```

69     bytesRead = clientStream.Read(message, 0, 4096);
70     }
71     catch
72     {
73         //a socket error has occurred
74         break;
75     }
76
77     if (bytesRead == 0)
78     {
79         //the client has disconnected from the server
80         connectedClients--;
81         lblNumberOfConnections.Text = connectedClients.ToString();
82         break;
83     }
84
85     //message has successfully been received
86     ASCIIEncoding encoder = new ASCIIEncoding();
87
88     // Convert the Bytes received to a string and display it on the Se
89     rver Screen
90     string msg = encoder.GetString(message, 0, bytesRead);
91     WriteMessage(msg);
92
93     // Now Echo the message back
94
95     Echo(msg, encoder, clientStream);
96 }
97 tcpClient.Close();
98 }
99
100 private void WriteMessage(string msg)
101 {
102     if (this.rtbServer.InvokeRequired)
103     {
104         WriteMessageDelegate d = new WriteMessageDelegate(WriteMessage);
105         this.rtbServer.Invoke(d, new object[] { msg });
106     }
107     else
108     {
109         this.rtbServer.AppendText(msg + Environment.NewLine);
110     }
111 }
112
113 /// <summary>
114 /// Echo the message back to the sending client
115 /// </summary>
116 /// <param name="msg">
117 /// String: The Message to send back
118 /// </param>
119 /// <param name="encoder">
120 /// Our ASCIIEncoder
121 /// </param>
122 /// <param name="clientStream">
123 /// The Client to communicate to
124 /// </param>
125 private void Echo(string msg, ASCIIEncoding encoder, NetworkStream
126 clientStream)
127 {
128     // Now Echo the message back
129     byte[] buffer = encoder.GetBytes(msg);
130
131     clientStream.Write(buffer, 0, buffer.Length);
132     clientStream.Flush();
133 }
134 }

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Windows.Forms;
5
6  namespace ServerApp
7  {
8      static class Program
9      {
10         /// <summary>
11         /// The main entry point for the application.
12         /// </summary>
13         [STAThread]
14         static void Main()
15         {
16             Application.EnableVisualStyles();
17             Application.SetCompatibleTextRenderingDefault(false);
18             Application.Run(new Form1());
19         }
20     }
21 }
```


- A**
AcceptTcpClient, 6
AppendText, 7
Application, 8
ASCIIEncoding, 7
- B**
buffer, 7
bytesRead, 6, 7
- C**
client, 6
clientStream, 6, 7
clientThread, 6
Close, 7
Collections, 6, 8
ComponentModel, 6
connectedClients, 6, 7
- D**
d, 7
Data, 6
Drawing, 6
- E**
Echo, 7
EnableVisualStyles, 8
encoder, 7
Environment, 7
- F**
Flush, 7
Form, 6
Form1, 6, 8
Forms, 6, 8
- G**
Generic, 6, 8
GetBytes, 7
GetStream, 6
GetString, 7
- H**
HandleClientComm, 6
- I**
InitializeComponent, 6
Invoke, 7
InvokeRequired, 7
IPAddress, 6
- L**
lblNumberOfConnections, 6, 7
Length, 7
Linq, 6, 8
ListenForClients, 6
listenThread, 6
Loopback, 6
- M**
Main, 8
message, 6, 7
msg, 6, 7
- N**
Net, 6
NetworkStream, 6, 7
NewLine, 7
- P**
ParameterizedThreadStart, 6
partial, 6
Program, 8
- R**
Read, 7
rtbServer, 7
Run, 8
- S**
Server, 6
ServerApp, 6, 8
SetCompatibleTextRenderingDefault, 8
Sockets, 6
Start, 6
STAThread, 8
System, 6, 8
- T**
tcpClient, 6, 7
TcpClient, 6
tcpListener, 6
TcpListener, 6
Text, 6, 7
Thread, 6
Threading, 6
ThreadStart, 6
ToString, 6, 7
- W**
Windows, 6, 8
Write, 7
WriteMessage, 7
WriteMessageDelegate, 6, 7