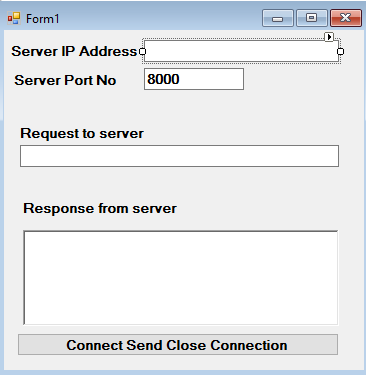
**Simple Client Server Apps Example**

Code for the server: this a console app server. You should be able to turn it into a windows app  
by putting the code in Main inside a Task, within a button click.

This server takes few commands like add sub mul div. the server performs the functions and returns results back to the client

|  |
| --- |
| namespace SimpleConsoleServer  {  /\*  Building a simple client server application. Allowing 2 computers to communicate through sockets.  When 2 or more computers communicate together, one of the computer is the server and the others are the clients.    A server application needs to be installed on the server computer  A client application needs to be installed ont the client computers    A computer may have both the server and the client apps and have both application communicate within the  same computer    \*/  class Program  {  static void Main(string[] args)  {  const int port = 8000;  Socket server = null;  try  {  //1. Create a Socket object  server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  //2. Create an IP EndPoint (combination of IP Address and Port number)  IPHostEntry hostEntry = Dns.GetHostEntry(Dns.GetHostName());  IPAddress[] ipaddresses = hostEntry.AddressList;  foreach (IPAddress address in ipaddresses)  Console.WriteLine(address);  //IPEndPoint endP = new IPEndPoint(IPAddress.Any, port);  IPEndPoint endP = new IPEndPoint(IPAddress.Any, port);  //3. Bind the IPEndPoint to the Socket  server.Bind(endP);  //4. Set the socket to listen for incoming requests (save client requests to  // an internal queue)  server.Listen(20);  string hostname = Dns.GetHostName();  Console.WriteLine("\nServer: {0} is ready at ip: {1}:{2}",hostname,ipaddresses[1], port);  //5. within a loop. Accept or get client request from the queue  while (true)  {  //use the Accept method to get the next waiting client from the internal queue  //the Accept method blocks, that means that the Accept method waits until it finds a  //client in the queue and returns a reference to it.  Socket client = server.Accept();  //display client information  DisplayClientInfo(client);  //could log info to a logfile  //Process the client  Task.Factory.StartNew(() => ProcessClient(client));  }  }  catch (SocketException se)  {  Console.WriteLine("\nSocket Exception\n");  Console.WriteLine(se.Message);    }  Console.ReadLine();//pause  }  //method to display client information  static void DisplayClientInfo(Socket client)  {  //Get the client EndPoint using the RemoteEndPoint property of the Socket  IPEndPoint clientEndP = (IPEndPoint)client.RemoteEndPoint;  //client IP Address  IPAddress clientIPAddress = clientEndP.Address;  //client port  int clientPort = clientEndP.Port;  //get client hostname  string clientName = Dns.GetHostEntry(clientIPAddress).HostName;  Console.WriteLine("\nClient Request From {0} at {1}:{2}", clientName, clientIPAddress, clientPort);  }  //method to process a client  static void ProcessClient(Socket client)  {  //1. Receive client request  byte[] buffer = new byte[256];  int bytesReceived = client.Receive(buffer);  //convert bytes to a string (convert only the bytes received not 256 bytes set up  string request = Encoding.UTF8.GetString(buffer, 0, bytesReceived);  //2. Do some work (process it)  string response = ProcessRequest(request);  //3. return a response  SendResponse(client, response);  }  static string ProcessRequest(string request)  {  if (request == String.Empty)  return "Invalid Request";  string response = "";  try  {  string[] words = request.Split(new char[] { ' ', ',' }, StringSplitOptions.RemoveEmptyEntries);  if (words.Length != 3)  {  return "Invalid Request";  }  double v1 = double.Parse(words[1]);  double v2 = double.Parse(words[2]);    switch (words[0].ToLower())  {  case "add":  double result = v1 + v2;  response = result.ToString();  break;  case "sub":  result = v1 - v2;  response = result.ToString();  break;  case "mul":  result = v1 \* v2;  response = result.ToString();  break;  case "div":  result = v1 / v2;  response = result.ToString();  break;  case "test":  string filepath = Directory.GetCurrentDirectory();  int index = filepath.LastIndexOf("\\");  string filename = filepath.Remove(index);  index = filename.LastIndexOf("\\");  filename = filename.Remove(index+1);  filename += "Program.cs";  response = File.ReadAllText(filename);  //response = filename;  break;  default:  response = "Invalid Command";  break;  }  }  catch (FormatException fe)  {  response = "Invalid data";  }  return response;  }  //send response back to client  static void SendResponse(Socket client, string response)  {  try  {  //convert response string to byte array  byte[] buffer = Encoding.UTF8.GetBytes(response);  client.Send(buffer);  }  catch (SocketException se)  {  }  finally  {  client.Shutdown(SocketShutdown.Both);  client.Close();  }  }  }  }  ///Lab assignment:  ///Allow the server to accept command "lottery"  ///When the server receives this command it is to return a string with 6 distinct numbers from 1 to 49  ///  ///Allow server to accept command "Power" followed by 2 values: base and exponent. The server is to return a value  ///equal to 'base to the power of exponent.  ///Example Power 2 5 should cause the server to return 32 |

Client



|  |
| --- |
| using System.Net;  using System.Net.Sockets;  namespace SimpleWindowsClient  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void btnSendRequest\_Click(object sender, EventArgs e)  {  Socket client = null;  try  {  string request = txtRequest.Text;  if (request.Trim() == String.Empty)  {  MessageBox.Show("Must enter a request");  return;  }  //1. create sockect  client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);  //2. build the server endpoint  IPAddress ipaddress = IPAddress.Parse(txtServerIPAddress.Text);  int port = int.Parse(txtServerPortNo.Text);  IPEndPoint serverEndP = new IPEndPoint(ipaddress, port);  //3.Connect to the server, using the connect method defined in the Socket class  client.Connect(serverEndP);  //Request/Response client server app  //send request  //convert to bytes  byte[] data = Encoding.UTF8.GetBytes(request);  client.Send(data);  //receive response  byte[] buffer = new byte[256];  StringBuilder sb = new StringBuilder();  int bytesReceived;  while ((bytesReceived = client.Receive(buffer)) != 0)  {  string response = Encoding.UTF8.GetString(buffer, 0, bytesReceived);  sb.Append(response);  }  //int bytesReceived = client.Receive(buffer);  //string response = Encoding.UTF8.GetString(buffer, 0, bytesReceived);  //display it  rtbResponse.Text = "Result = " + sb.ToString();  }  catch (SocketException se)  {  MessageBox.Show(se.Message);  }  catch (FormatException fe)  {  MessageBox.Show(fe.Message);  }  finally  {  if (client != null && client.Connected)  {  client.Shutdown(SocketShutdown.Both);  client.Close();  }  }  }  }  } |