ChatPlatform Reference

Table of Contents

ChatPlatform Reference

Namespaces

ChatClient 5, ChatPlatform 12, ChatPlatformUnitTest 24

ChatClient Namespace

This namespace holds all the references to the client application.

Classes

Client5, ClientHandler5, MessageRecievedEventArgs9

Enumerations

MESSAGE_TYPE11

Client Class

This class is what contains the main execution method.

System.Object
  ChatClient.Client

C#

```
public class Client
```

Requirements

Namespace: ChatClient5

Assembly: ChatClient (in ChatClient.exe)

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

ClientHandler Class

This class manages the connection to the server. It is contained in its own object so that multiple can be dynamically spawned if more than one client connection is needed.

System.Object
  ChatClient.ClientHandler

C#

```
public class ClientHandler
```

Requirements

Namespace: ChatClient5

Assembly: ChatClient (in ChatClient.exe)

Constructors

ClientHandler6

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), PrintMessage7, ReceiveMessageFromServer7, SendLoginMessage8, SendMessage8, StopClient9, ToString (inherited from Object)

Events

ChatRecievedEventHandler9

ClientHandler Constructor

This constructor creates an instance of the ClientHandler.

C#

```
public ClientHandler(
   string address,
   int port,
   string username
)
```

Parameters

address

The IP Address of the server

port

The port the server is running on

username

The username of the client

Source code

```
 public ClientHandler(string address, Int32 port, string username)
 {
     client = new TcpClient(address, port);
     stream = client.GetStream();
     this.username = username;

     //Subscribes incoming events to the PrintMessage delegate.
     ChatRecievedEventHandler += PrintMessage;

     //Sends the login message to the server so it knows what the client
username is
     SendLoginMessage();
     //Spawn a thread to wait for messages
     Thread t = new Thread(new ThreadStart(ReceiveMessageFromServer));
     t.Start();
 }
```

See Also

Applies to: ClientHandler5

ClientHandler.PrintMessage Method

This message allows the ChatRecievedEventHandler to print to the output window

C#

```
public void PrintMessage(
   object sender,
   EventArgs e
)
```

Parameters

sender

ClientHander responsible to invoking the ChatRecievedEventHandler event

e

The MessageRecievedEventArgs sent by invoking the event

Source code

```
public void PrintMessage(object sender, EventArgs e)
{
    MessageRecievedEventArgs m = e as MessageRecievedEventArgs;
    Console.WriteLine(m.message);
}
```

See Also

Applies to: ClientHandler5

ClientHandler.ReceiveMessageFromServer Method

This loop runs on a separate thread so that the client can recieve messages that other clients have sent.

C#

```
public void ReceiveMessageFromServer()
```

Source code

```
 public void ReceiveMessageFromServer()
 {
     while(true)
     {
         Byte[] data = new byte[256];
         if (stream.Read(data, 0, data.Length) != 0)
         {
             ChatRecievedEventHandler?.Invoke(this, new
MessageRecievedEventArgs(Encoding.ASCII.GetString(data).Replace("\0", "")));
         }
     }
 }
```

See Also

Applies to: ClientHandler5

ClientHandler.SendLoginMessage Method

Sends the login message to the server so it knows what the client username is.

C#

```
public void SendLoginMessage()
```

Source code

```
 public void SendLoginMessage()
 {
     Byte[] data = System.Text.Encoding.ASCII.GetBytes(username + ":" + "" +
"//" + MESSAGE_TYPE.LOGIN);
     stream.Write(data, 0, data.Length);
 }
```

See Also

Applies to: ClientHandler5

ClientHandler.SendMessage Method

Sends a message to the server.

C#

```
public void SendMessage(
   string s
)
```

Parameters

s

The message being sent

Source code

```
 public void SendMessage(string s)
 {
     Byte[] data = System.Text.Encoding.ASCII.GetBytes(username + ":" + s + "//"
+ MESSAGE_TYPE.MESSAGE_SENT);
     stream.Write(data, 0, data.Length);
 }
```

See Also

Applies to: ClientHandler5

ClientHandler.StopClient Method

This method closes the TCP and Network streams.

C#

```
public void StopClient()
```

Source code

```
public void StopClient()
{
    stream.Close();
    client.Close();
}
```

See Also

Applies to: ClientHandler5

ChatRecievedEventHandler Event

This event handler keeps track of incoming messages and allows methods, like the PrintMessage() method to hook into the event called when a message is recieved.

C#

```
public event EventHandler ChatRecievedEventHandler
```

Source code

```
public event EventHandler ChatRecievedEventHandler;
```

See Also

Applies to: ClientHandler5

MessageRecievedEventArgs Class

This EventArgs class is derived so that the client can trigger an event containing a message.

System.Object
  System.EventArgs
    ChatClient.MessageRecievedEventArgs

C#

```
public class MessageRecievedEventArgs : EventArgs
```

Requirements

Namespace: ChatClient5

Assembly: ChatClient (in ChatClient.exe)

Constructors

MessageRecievedEventArgs10

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

message10

MessageRecievedEventArgs Constructor

C#

```
public MessageRecievedEventArgs(
   string message
)
```

Parameters

message

Source code

```
public MessageRecievedEventArgs(string message)
{
    this.message = message;
}
```

See Also

Applies to: MessageRecievedEventArgs9

message Field

C#

```
public string message
```

Source code

```
public string message;
```

See Also

Applies to: MessageRecievedEventArgs9

MESSAGE_TYPE Enumeration

There are three kinds of messages sent to the server. Each one is encoded in the string sent to the server.

Constant	Value	Description
DISCONNECT	1	Lets the server know that the client is disconnecting.
LOGIN	0	Login message tells the server the username of the new client.
MESSAGE_SENT	2	Lets the server know that a message intended for rebroadcast has been sent.

Requirements

Namespace: ChatClient5

Assembly: ChatClient (in ChatClient.exe)

ChatPlatform Namespace

Classes

ConnectionHandler12, MessageRecievedEventArgs15, ServerHandler17

Enumerations

MESSAGE_TYPE23

ConnectionHandler Class

This class holds the information for incoming server connections.

System.Object
  ChatPlatform.ConnectionHandler

C#

```
public class ConnectionHandler
```

Requirements

Namespace: ChatPlatform12

Assembly: ChatPlatform (in ChatPlatform.exe)

Constructors

ConnectionHandler12

Properties

Username13

Methods

AwaitData14, Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), SendMeMessage14, ToString (inherited from Object)

ConnectionHandler Constructor

This constructor creates a new object that holds the information of a client application.

C#

```
public ConnectionHandler(
   TcpClient client,
   string name,
   EventHandler chatEventHandler,
   UInt32 bufferSize
)
```

Parameters

client

The incoming TcpClient pulled from the AcceptTcpClient method

name

The username of the incoming connection

chatEventHandler

The event handler for writing to the console

bufferSize

Desired buffer size for incoming data

Source code

```
 public ConnectionHandler(TcpClient client, string name, EventHandler
chatEventHandler, uint bufferSize)
 {
     bytes = new byte[bufferSize];
     this.chatEventHandler = chatEventHandler;
     this.name = name;

     this.client = client;
     stream = client.GetStream();
 }
```

See Also

Applies to: ConnectionHandler12

ConnectionHandler.Username Property

Public accessor for the username of the connection

C#

```
public string Username {get; set;}
```

Source code

```
 public string Username
 {
     get
     {
         return name;
     }
     set
     {
         name = value;
     }
 }
```

See Also

Applies to: ConnectionHandler12

ConnectionHandler.AwaitData Method

This loop waits for incoming data

C#

```
public void AwaitData()
```

Source code

```
 public void AwaitData()
 {
     try
     {
         int i;
         while ((i = stream.Read(bytes, 0, bytes.Length)) != 0)
         {
             //Incoming data is stored in buffer
             data = System.Text.Encoding.ASCII.GetString(bytes, 0, i);

             //Username, message_type, and message being parsed from the data
             string name = data.Substring(0, data.IndexOf(':'));
             MESSAGE_TYPE messageType =
(MESSAGE_TYPE)Enum.Parse(typeof(MESSAGE_TYPE),
data.Substring(data.IndexOf("//")+2));
             string message = data.Substring(data.IndexOf(':')+1,
data.IndexOf("//") - data.IndexOf(':') - 1);

             //Calls an event to write to the console
             chatEventHandler?.Invoke(this, new MessageRecievedEventArgs(name,
messageType, message));
         }
     }
     catch (System.IO.IOException)
     {
         ServerHandler.DisconnectClient(this);
     }
     catch (Exception)
     {
         Console.WriteLine("Error Occurred");
         ServerHandler.DisconnectClient(this);
     }
 }
```

See Also

Applies to: ConnectionHandler12

ConnectionHandler.SendMeMessage Method

Sends a message back to the client

C#

```
public void SendMeMessage(
   byte[] message
)
```

Parameters

message

The message being sent

Source code

```
public void SendMeMessage(Byte[] message)
{
    stream.Write(message, 0, message.Length);
}
```

See Also

Applies to: ConnectionHandler12

MessageRecievedEventArgs Class

This EventArgs class is derived so that the client can trigger an event containing a username, message type, and message.

System.Object
  System.EventArgs
    ChatPlatform.MessageRecievedEventArgs

C#

```
public class MessageRecievedEventArgs : EventArgs
```

Requirements

Namespace: ChatPlatform12

Assembly: ChatPlatform (in ChatPlatform.exe)

Constructors

MessageRecievedEventArgs16

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

message16, sender16, t17

MessageRecievedEventArgs Constructor

C#

```
public MessageRecievedEventArgs(
   string sender,
   MESSAGE_TYPE t,
   string message
)
```

Parameters

sender

t

message

Source code

```
public MessageRecievedEventArgs(string sender, MESSAGE_TYPE t, string message)
{
    this.sender = sender;
    this.message = message;
    this.t = t;
}
```

See Also

Applies to: MessageRecievedEventArgs15

message Field

C#

```
public string message
```

Source code

```
public string message;
```

See Also

Applies to: MessageRecievedEventArgs15

sender Field

C#

```
public string sender
```

Source code

```
public string sender;
```

See Also

Applies to: MessageRecievedEventArgs15

t Field

C#

```
public MESSAGE_TYPE t
```

Source code

```
public MESSAGE_TYPE t;
```

See Also

Applies to: MessageRecievedEventArgs15

ServerHandler Class

This class holds all the information for the server. It's contained in its own class so that multiple instances can be spawned if need be.

System.Object
  ChatPlatform.ServerHandler

C#

```
public static class ServerHandler
```

Requirements

Namespace: ChatPlatform12

Assembly: ChatPlatform (in ChatPlatform.exe)

Methods

BeginAcceptConnections18, Broadcast18, DisconnectClient19, Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), IsReady19, MemberwiseClone (inherited from Object), RecieveMessage20, Start21, Stop22, ToString (inherited from Object)

Events

ChatEventHandler22

Fields

ClientList22

ServerHandler.BeginAcceptConnections Method

Allows the server to start accepting connetions

C#

```
public static void BeginAcceptConnections()
```

Source code

```
public static void BeginAcceptConnections()
{
    if (!acceptRunning)
    {
        Thread t = new Thread(new ThreadStart(PrivateBeginAcceptConnections));
        t.Start();
    }
}
```

See Also

Applies to: ServerHandler17

ServerHandler.Broadcast Method

This method sends a message back to all clients except the client that sent the message

C#

```
public static void Broadcast(
   ConnectionHandler sender,
   string message
)
```

Parameters

sender

The client that sent the message

message

The message being sent

Source code

```
public static void Broadcast(ConnectionHandler sender, string message)
{
    foreach(ConnectionHandler c in ClientList)
    {
        if (!c.Equals(sender))
        {
            Byte[] data = System.Text.Encoding.ASCII.GetBytes(message);
            c.SendMeMessage(data);
        }
    }
}
```

See Also

Applies to: ServerHandler17

ServerHandler.DisconnectClient Method

This methods ensure that the disconnecting client is removed from the list.

C#

```
public static void DisconnectClient(
   ConnectionHandler h
)
```

Parameters

h

Client that is disconnecting

Source code

```
 public static void DisconnectClient(ConnectionHandler h)
 {
     ChatEventHandler?.Invoke(h, new MessageRecievedEventArgs(h.Username,
MESSAGE_TYPE.DISCONNECT, ""));
     ClientList.Remove(h);
 }
```

See Also

Applies to: ServerHandler17

ServerHandler.IsReady Method

See if the server is ready to start

C#

```
public static bool IsReady()
```

Source code

```
public static bool IsReady()
{
    return isReady;
}
```

See Also

Applies to: ServerHandler17

ServerHandler.RecieveMessage Method

Method called when the ChatEventHandler event is invoked

C#

```
public static void RecieveMessage(
   object sender,
   EventArgs e
)
```

Parameters

sender

The ConnectionHandler responsible for invoking the method

e

The arguments containing the username, message type, and message

Source code

```
public static void RecieveMessage(object sender, EventArgs e)
{
    MessageRecievedEventArgs m = e as MessageRecievedEventArgs;
    ConnectionHandler c = sender as ConnectionHandler;

    switch (m.t)
    {
        case MESSAGE_TYPE.LOGIN:
            c.Username = m.sender;
            Console.WriteLine(c.Username + " connected.");
            Broadcast(c, c.Username + " connected.");
            break;
        case MESSAGE_TYPE.MESSAGE_SENT:
            Console.WriteLine(m.sender + ": " + m.message);
            Broadcast(c, c.Username + ": " + m.message);
            break;
        case MESSAGE_TYPE.DISCONNECT:
            Console.WriteLine(m.sender + " disconnected.");
            Broadcast(c, c.Username + " disconnected.");
            break;
        default:
            break;
    }

    Console.WriteLine(ClientList.Count);
}
```

See Also

Applies to: ServerHandler17

ServerHandler.Start Method

Instantiates all required objects to run a server

C#

```
public static bool Start(
   int port
)
```

Parameters

port

Startup port

Returns

Returns true if the server was sucessfully setup

Source code

```
public static bool Start(int port)
{
    // Subscribe the Event handler to the receive message function.
    ChatEventHandler += RecieveMessage;

    // Attempt to start a new TCP Listener server.
    try
    {
        server = new TcpListener(IPAddress.Any, port);
        server.Start();

        // Set ready flag to true.
        isReady = true;
        return true;
    }
    catch(Exception e)
    {
        return false;
    }
}
```

See Also

Applies to: ServerHandler17

ServerHandler.Stop Method

Closes the server

C#

```
public static void Stop()
```

Source code

```
 public static void Stop()
 {
     server.Stop();
 }
```

See Also

Applies to: ServerHandler17

ChatEventHandler Event

Handles all incoming messages

C#

```
public event EventHandler ChatEventHandler
```

Source code

```
 public static event EventHandler ChatEventHandler;
```

See Also

Applies to: ServerHandler17

ClientList Field

Holds a list of connected clients

C#

```
new public static List<ConnectionHandler> ClientList
```

Source code

```
 public static List<ConnectionHandler> ClientList = new
List<ConnectionHandler>();
```

See Also

Applies to: ServerHandler17

MESSAGE_TYPE Enumeration

There are three kinds of messages sent to the server. Each one is encoded in the string sent to the server.

Constant	Value	Description
DISCONNECT	1	Lets the server know that the client is disconnecting.
LOGIN	0	Login message tells the server the username of the new client.
MESSAGE_SENT	2	Lets the server know that a message intended for rebroadcast has been sent.

Requirements

Namespace: ChatPlatform12

Assembly: ChatPlatform (in ChatPlatform.exe)

ChatPlatformUnitTest Namespace

Classes

UnitTest1 24

UnitTest1 Class

System.Object
  ChatPlatformUnitTest.UnitTest1

C#

```
[TestClass()]
public class UnitTest1
```

Requirements

Namespace: ChatPlatformUnitTest 24

Assembly: ChatPlatformUnitTest (in ChatPlatformUnitTest.dll)

Methods

CheckServerStartup 24, ConnectionTest 25, Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), TestClientNumber 25, TestLogin 26, TestTalkback 27, ToString (inherited from Object)

UnitTest1.CheckServerStartup Method

C#

```
[TestMethod()]
public void CheckServerStartup()
```

Source code

```
 [TestMethod]
 public void CheckServerStartup()
 {
     Assert.IsTrue(ChatPlatform.ServerHandler.Start(13000), "Server Unable to
Start");
     ChatPlatform.ServerHandler.Stop();
 }
```

See Also

Applies to: UnitTest1 24

UnitTest1.ConnectionTest Method

C#

```
[TestMethod()]
public void ConnectionTest()
```

Source code

```
 [TestMethod]
 public void ConnectionTest()
 {
     List<ChatPlatform.MessageRecievedEventArgs> list = new
List<ChatPlatform.MessageRecievedEventArgs>();

     ChatPlatform.ServerHandler.ChatEventHandler += delegate(object sender,
EventArgs eventArgs)
     {
         ChatPlatform.MessageRecievedEventArgs m = eventArgs as
ChatPlatform.MessageRecievedEventArgs;
         list.Add(m);
     };

     ChatPlatform.ServerHandler.Start(13000);
     ChatPlatform.ServerHandler.BeginAcceptConnections();
     Thread.Sleep(1000);

     ChatClient.ClientHandler c = new ChatClient.ClientHandler("127.0.0.1",
13000, "1");

     Thread.Sleep(1000);
     c.SendMessage("Test");
     Thread.Sleep(1000);

     Debug.WriteLine("Messages Recieved");
     foreach (ChatPlatform.MessageRecievedEventArgs m in list)
         Debug.WriteLine("{0}: {1} //{2}", m.sender, m.message, m.t);

     Assert.IsTrue(list[1].message.Equals("Test"));

     c.StopClient();
     ChatPlatform.ServerHandler.Stop();
 }
```

See Also

Applies to: UnitTest1 24

UnitTest1.TestClientNumber Method

C#

```
[TestMethod()]
public void TestClientNumber()
```

Source code

```
 [TestMethod]
 public void TestClientNumber()
 {
     List<ChatClient.ClientHandler> list = new List<ChatClient.ClientHandler>();

     ChatPlatform.ServerHandler.Start(13000);
     ChatPlatform.ServerHandler.BeginAcceptConnections();
     Thread.Sleep(1000);

     try
     {
         while (list.Count < 1000)
         {
             ChatClient.ClientHandler c1 = new
ChatClient.ClientHandler("127.0.0.1", 13000, "TESTUSER1");
             list.Add(c1);
         }
     }
     catch(Exception e)
     {
         Debug.WriteLine(e.Message);
     }


 }
```

See Also

Applies to: UnitTest1 24

UnitTest1.TestLogin Method

C#

```
[TestMethod()]
public void TestLogin()
```

Source code

```
 [TestMethod]
 public void TestLogin()
 {
     ChatPlatform.ServerHandler.Start(13000);
     ChatPlatform.ServerHandler.BeginAcceptConnections();
     Thread.Sleep(1000);

     ChatClient.ClientHandler c = new ChatClient.ClientHandler("127.0.0.1",
13000, "TESTUSER");
     Thread.Sleep(1000);
Assert.IsTrue(ChatPlatform.ServerHandler.ClientList[0].Username.Equals("TESTUSER
"));

     c.StopClient();
     ChatPlatform.ServerHandler.Stop();
 }
```

See Also

Applies to: UnitTest1 24

UnitTest1.TestTalkback Method

C#

```
[TestMethod()]
public void TestTalkback()
```

Source code

```
 [TestMethod]
 public void TestTalkback()
 {
     List<ChatClient.MessageRecievedEventArgs> list = new
List<ChatClient.MessageRecievedEventArgs>();

     ChatPlatform.ServerHandler.Start(13000);
     ChatPlatform.ServerHandler.BeginAcceptConnections();
     Thread.Sleep(1000);

     ChatClient.ClientHandler c1 = new ChatClient.ClientHandler("127.0.0.1",
13000, "TESTUSER1");
     ChatClient.ClientHandler c2 = new ChatClient.ClientHandler("127.0.0.1",
13000, "TESTUSER2");
     Thread.Sleep(1000);

     c2.ChatRecievedEventHandler += delegate (object sender, EventArgs
eventArgs)
     {
         ChatClient.MessageRecievedEventArgs m = eventArgs as
ChatClient.MessageRecievedEventArgs;
         list.Add(m);
     };

     c1.SendMessage("TESTMESSAGE");
     Thread.Sleep(1000);

     foreach(ChatClient.MessageRecievedEventArgs m in list)
         Debug.WriteLine("{0}", m.message);

     Assert.IsTrue("TESTUSER1: TESTMESSAGE".Equals(list[0].message));
```

```
    c1.StopClient();
    c2.StopClient();
    ChatPlatform.ServerHandler.Stop();
}
```

See Also

Applies to: UnitTest1 24

Index