**Asynchronous Server Socket Example**

**.NET Framework 4**

The following example program creates a server that receives connection requests from clients. The server is built with an asynchronous socket, so execution of the server application is not suspended while it waits for a connection from a client. The application receives a string from the client, displays the string on the console, and then echoes the string back to the client. The string from the client must contain the string "<EOF>" to signal the end of the message.

**C#**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

// State object for reading client data asynchronously

public class StateObject {

// Client socket.

public Socket workSocket = null;

// Size of receive buffer.

public const int BufferSize = 1024;

// Receive buffer.

public byte[] buffer = new byte[BufferSize];

// Received data string.

public StringBuilder sb = new StringBuilder();

}

public class AsynchronousSocketListener {

// Thread signal.

public static ManualResetEvent allDone = new ManualResetEvent(false);

public AsynchronousSocketListener() {

}

public static void StartListening() {

// Data buffer for incoming data.

byte[] bytes = new Byte[1024];

// Establish the local endpoint for the socket.

// The DNS name of the computer

// running the listener is "host.contoso.com".

IPHostEntry ipHostInfo = Dns.Resolve(Dns.GetHostName());

IPAddress ipAddress = ipHostInfo.AddressList[0];

IPEndPoint localEndPoint = new IPEndPoint(ipAddress, 11000);

// Create a TCP/IP socket.

Socket listener = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp );

// Bind the socket to the local endpoint and listen for incoming connections.

try {

listener.Bind(localEndPoint);

listener.Listen(100);

while (true) {

// Set the event to nonsignaled state.

allDone.Reset();

// Start an asynchronous socket to listen for connections.

Console.WriteLine("Waiting for a connection...");

listener.BeginAccept(

new AsyncCallback(AcceptCallback),

listener );

// Wait until a connection is made before continuing.

allDone.WaitOne();

}

} catch (Exception e) {

Console.WriteLine(e.ToString());

}

Console.WriteLine("\nPress ENTER to continue...");

Console.Read();

}

public static void AcceptCallback(IAsyncResult ar) {

// Signal the main thread to continue.

allDone.Set();

// Get the socket that handles the client request.

Socket listener = (Socket) ar.AsyncState;

Socket handler = listener.EndAccept(ar);

// Create the state object.

StateObject state = new StateObject();

state.workSocket = handler;

handler.BeginReceive( state.buffer, 0, StateObject.BufferSize, 0,

new AsyncCallback(ReadCallback), state);

}

public static void ReadCallback(IAsyncResult ar) {

String content = String.Empty;

// Retrieve the state object and the handler socket

// from the asynchronous state object.

StateObject state = (StateObject) ar.AsyncState;

Socket handler = state.workSocket;

// Read data from the client socket.

int bytesRead = handler.EndReceive(ar);

if (bytesRead > 0) {

// There might be more data, so store the data received so far.

state.sb.Append(Encoding.ASCII.GetString(

state.buffer,0,bytesRead));

// Check for end-of-file tag. If it is not there, read

// more data.

content = state.sb.ToString();

if (content.IndexOf("<EOF>") > -1) {

// All the data has been read from the

// client. Display it on the console.

Console.WriteLine("Read {0} bytes from socket. \n Data : {1}",

content.Length, content );

// Echo the data back to the client.

Send(handler, content);

} else {

// Not all data received. Get more.

handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,

new AsyncCallback(ReadCallback), state);

}

}

}

private static void Send(Socket handler, String data) {

// Convert the string data to byte data using ASCII encoding.

byte[] byteData = Encoding.ASCII.GetBytes(data);

// Begin sending the data to the remote device.

handler.BeginSend(byteData, 0, byteData.Length, 0,

new AsyncCallback(SendCallback), handler);

}

private static void SendCallback(IAsyncResult ar) {

try {

// Retrieve the socket from the state object.

Socket handler = (Socket) ar.AsyncState;

// Complete sending the data to the remote device.

int bytesSent = handler.EndSend(ar);

Console.WriteLine("Sent {0} bytes to client.", bytesSent);

handler.Shutdown(SocketShutdown.Both);

handler.Close();

} catch (Exception e) {

Console.WriteLine(e.ToString());

}

}

public static int Main(String[] args) {

StartListening();

return 0;

}

}