Basics of working with pipes in C# .NET part 3: message transmission

JUNE 19, 2015 LEAVE A COMMENT (HTTPS://DOTNETCODR.COM/2015/06/19/BASICS-OF-WORKING-WITH-PIPES-IN-C-NET-PART-3-MESSAGE-TRANSMISSION/#RESPOND)

So far in the posts on pipe streams in .NET we've been considering the transmission of single bytes. That's not really enough for a real-world messaging application. In this post we'll extend our discussion to complete messages.

We have to explicitly indicate that we want to process messages in the NamedPipeServerStream constructor. Also, we'll need to read the messages in chunks. A message is represented by a byte array and we don't know in advance how long the message will be. Fortunately the NamedPipeServerStream object has an IsMessageComplete property which we'll be able to use in the do-while loop:

```
1  private static void ReceiveSingleMessageFromClient()
2  {
3      using (NamedPipeServerStream namedPipeServer = new NamedPipeServerStream("test-pipe", PipeDirecti
4          1, PipeTransmissionMode.Message))
5      {
6          namedPipeServer.WaitForConnection();
7          StringBuilder messageBuilder = new StringBuilder();
8          string messageChunk = string.Empty;
9          byte[] messageBuffer = new byte[5];
10         do
11         {
12             namedPipeServer.Read(messageBuffer, 0, messageBuffer.Length);
13             messageChunk = Encoding.UTF8.GetString(messageBuffer);
14             messageBuilder.Append(messageChunk);
15             messageBuffer = new byte[messageBuffer.Length];
16         }
17         while (!namedPipeServer.IsMessageComplete);
18
19         Console.WriteLine(messageBuilder.ToString());
20     }
21 }
```

The client side is much less code. You can type a message in the console which will be sent to the server as a byte array:

```
1  private static void SendSingleMessageToServer()
2  {
3      using (NamedPipeClientStream namedPipeClient = new NamedPipeClientStream("test-pipe"))
4      {
5          Console.Write("Enter a message to be sent to the server: ");
6          string message = Console.ReadLine();
7          namedPipeClient.Connect();
8          byte[] messageBytes = Encoding.UTF8.GetBytes(message);
9          namedPipeClient.Write(messageBytes, 0, messageBytes.Length);
10     }
11 }
```

Call these methods from the client and server Main methods respectively. Start the server and client console apps separately, send a message from the client and the server's console window should print it similar to the following:

(https://dotnetcodr.files.wordpress.com/2015/03/single-message-sent-from-client-to-server-through-pipe.png)

Read the next part here (https://dotnetcodr.com/2015/06/23/basics-of-working-with-pipes-in-c-net-part-4-basic-conversation-with-messages/).

View the list of posts on Messaging here (https://dotnetcodr.com/messaging/).

FILED UNDER .NET, MESSAGING TAGGED WITH C#, MESSAGING, PIPE

About Andras Nemes

I'm a .NET/Java developer living and working in Stockholm, Sweden.