

Re: TCP listener

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-07/msg02130.html>

- *From:* "Markgoldin" <markgoldin_2000@xxxxxxxxxx>
 - *Date:* Fri, 18 Jul 2008 16:01:49 -0500
-

<TcpClient.Available property

Yes, I am working on that and on other comments that have been made here.

< how do you expect to exit out of the "go" loop?

This loop is required because of the main purpose of the program I am trying to adapt.

This is a push server. This product can communicate with .Net code and push data to the browser from it.

Since my main coding environment is not .Net I am trying to talk to .Net piece to provide data to be pushed.

One of ideas is to use TCP sockets for that.

So, I am using the push server product's .Net sample and trying to modify it to accept external data.

Like I said, it basically works. But now when I am testing the whole thing I have noticed

that it only works until I close connection to the c# listener.

"Peter Duniho" <NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx> wrote in message news:op.ueh84fh58jd0ej@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

On Fri, 18 Jul 2008 13:07:41 -0700, Markgoldin <markgoldin_2000@xxxxxxxxxx> wrote:

Ok, valid point.

```
private volatile bool go = true;
private volatile bool monitorPort = true;
```

```
byte[] byteReadStream = null;
TcpClient tcpc = DataAdapterLauncher.tcpl.AcceptTcpClient(); //accept
connection
while (go)
{
    while (monitorPort)
    {
        byteReadStream = new byte[tcpc.Available]; //allocate space for data
        tcpc.GetStream().Read(byteReadStream, 0, tcpc.Available);
        //read data into byte array
```

Re: TCP listener

```
if (byteReadStream != null)
{
    monitorPort = false;
}
System.Console.WriteLine("In the loop");
}
string str;
System.Text.ASCIIEncoding enc = new System.Text.ASCIIEncoding();
str = enc.GetString(byteReadStream);
monitorPort = true;
}
```

It works fine as long as a client keeps connection.

That's still not a complete code sample. And it raises more questions than it answers. For example:

— why do you have the "monitorPort" loop at all? assuming that's the literal code, "byteReadStream" is *_never_* going to be null, and so that loop will always execute the contained block exactly once before exiting

— why do you decode the bytes you read? you never use the "str" variable for anything once it's been assigned

— how do you expect to exit out of the "go" loop? the variable "go" is initialized to "true" and then never modified

— in what context does that code appear? what do you expect to happen when the connection is closed?

Also, as I mentioned before, you *_really_* should not be using the `TcpClient.Available` property. That said, if you insist, you definitely should not be passing it as the length parameter for the `Stream.Read()` method. It could change between the time you allocate the buffer and the time you call the `Read()` method. Just pass `byteReadStream.Length` instead.

Again, based on the code posted, I'm not surprised your program gets stuck in an endless, non-yielding loop. That's just what the code you posted looks like it would do. My guess is that if you fix your code to break out of the loop when the connection is closed, your problem would go away. But you would continue to have other the flaws in your code, and you should really be looking to fix those too.

Pete

.