

Re: simple Socket Programming question.

Re: simple Socket Programming question.

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2006-08/msg00922.html>

- *From:* "Francois Malgreve" <francois.malgreve@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 28 Aug 2006 16:56:51 +0700
-

Dear Barry,

Thanks a lot for the quick answer, I found myself finally doing nearly the same thing.

```
int numberOfBytesRead = stream.Read(data, 0, data.Length);
string returndata = System.Text.Encoding.ASCII.GetString(data, 0,
numberOfBytesRead);
```

```
while (numberOfBytesRead > 0)
{
numberOfBytesRead = stream.Read(data, 0, data.Length);
returndata += System.Text.Encoding.UTF8.GetString(data, 0,
numberOfBytesRead);
}
```

And you are right about the "using", I will implement it right away.

I don't have more than 1 concurrent download at a type then i don't need a asynchronous system. Nevertheless, my file downloaded can be pretty large and I would be really glad if you could point me to some information concerning the streaming strategy. Right now i put everything in a String as you can see in my code snippet and then write the string in a text file.

Thanks again,

Francois Malgreve.

You are ight

"Barry Kelly" <barry.j.kelly@xxxxxxxxxx> wrote in message <news:cpd5f21umo8t2gk0d7kp9j397gvei5npe2@xxxxxxxxxx>

Francois Malgreve wrote:

I am a bit new in socket programming and I have a question about what's the best way to implemnt a TCPClient that can receive long messages?

Re: simple Socket Programming question.

Re: simple Socket Programming question.

With blocking reads (i.e. you're not using `Begin*` and `End*`), you run in a loop reading from the stream / socket until either (1) something you've read tells you that the stream has ended, or (2) the read returns 0, zero bytes read.

For an example of (1), HTTP 1.1 with keep-alive requires a header in the response stream to tell the client how much data to expect. It looks like "Content-Length: <some number>\r\n". So, to read this kind of data, you'd read in the headers and parse them until the blank line in the HTTP response, then read that exact number of bytes in by looping around calling `Socket.Read()` or `NetworkStream.Read()`.

For an example of (2), see the code as I've modified it below:

```
// Receive the TcpServer.response.

// Buffer to store the response bytes.
data = new Byte[256];

// String to store the response ASCII representation.
String responseData = String.Empty;

MemoryStream buffer = new MemoryStream();
for (;;)
{
    int read = stream.Read(data, 0, data.Length);
    if (read == 0) // read of 0 bytes indicates end of stream.
        break;
    buffer.Write(data, 0, read);
}

// Now, buffer has all the data. You can convert it into an array
// with ToArray() if desired.

// Close everything.
stream.Close();
client.Close();
}
```

You should use 'using' to dispose of both the client and the stream. If an exception occurs in your existing code before these lines, the stream / client will never be closed.

For actual tactics that will work for long-running downloads of large amounts of data, you could look into (1) streaming the download to a

Re: simple Socket Programming question.

file on disk, and (2) getting a handle on how asynchronous sockets work to avoid lots of threads if you've got lots of concurrent downloads.

-- Barry

--

<http://barrkel.blogspot.com/>