.NET Framework アイデアノート

なにを使うか、どう使えるのか

第2回

秋月巌ソリューション事務所
秋月 巌 *AKIZUKI, Iwao*
http://www.akizuki.co.jp

WebサーバーとASP.NETでリアルタイムTCP/IPサーバーを開発する

Technology Tools

- ☑ Visual Basic .NET
- ❑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ☑ ASP.NET
- ☑ Internet Information Services
- ❑ Other:

Level

★ ★ ★ ☆ ☆

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、付録CD-ROMの¥DOTNET¥NOTEディレクトリに収録しています。

¥CHAT
クライアントプログラムとソースコード

¥ASPXCHAT
サーバープログラムとソースコード

Socketの知識なしでTCP/IPサーバーを開発できる

今回はIISとASP.NETを使って、汎用のTCP/IPサーバーアプリケーション開発をする方法を紹介する。まず、これがどういうことなのか、いかにこれを実現するのかについて、きちんと説明する必要がある。特にWebサーバーやASP.NETの動作を理解している人ほど、ここで紹介する技法は直感的に理解し難いだろう。

今回紹介するサンプルプログラムは、チャットサーバーとそのクライアント（図1、図2）である。動作は通常のチャットプログラムと同じと考えてもらいたい。では、なぜ、これをIISとASP.NETを使用して作成するかというと、そのほうが少ない学習量でマルチスレッドTCP/IPサーバーアプリケーションを開発できるからである。この記事を読み終えるころ、あなたはテキストベースの通信を行なうTCP/IPサーバーアプリケーションを作れるようになるだろう。

図1：［受信開始］ボタンのクリックで受信を開始

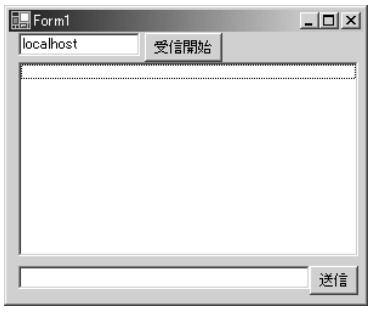

図2：テキストボックスに入力して［送信］ボタンをクリック、または［Enter］キーを押すとメッセージを送信

Webサーバーで リアルタイム応答アプリが作れる？

Webサーバーの知識がある人ならば、Webサーバーがサポートするプロトコルがステートレスであることを知っているはずである。そして、Webブラウザから行なわれた接続要求は1回の応答後に切断される。そのため、チャットサーバーのようにサーバーでイベントが発生したとき、すぐにクライアントに対してデータを送信するようなアプリケーションは作れないはずである。そのため、通常使われているWebチャットは一定間隔ごとにWebブラウザからサーバーにデータを要求する。つまり、リアルタイムでは動作しない。

しかし、ここで紹介するのは、ちゃんとリアルタイムで動作するチャットサーバーである（とはいえ厳密にいえばタイムラグはある）。他者が入力するとすぐに、その入力内容を受信することができる。つまり、Webチャットのように一定間隔ごとに更新されるのではない、普通のチャットプログラムである。

簡単に原理を説明すると、送信用と受信用の2つの接続を用いて、送信用の接続は送信後にすぐに接続を切断するが、受信用の接続はサーバーからクライアントに応答するまで、接続を切断せずに常に待機し続けるのである。他のユーザーが送ったメッセージを配信するときに一旦接続はサーバーによって切断されるが、クライアントはデータを受信したらもう一度すぐにサーバーに対して接続要求をする。そして再び、サーバーは接続を維持したまま待機し、他のクライアントがデータを送信してくるのを待つのである。誰かがメッセージを書き込むたびに、受信用の接続ではこのプロセスが繰り返される。一方、送信用の接続は、受信処理とは一切無関係に別スレッドで行なわれる。このスレッドでは、送信時のみWebサーバーに接続し、1回の送信処理が終了するたびに接続を切断する。これは通常のHTTPプロトコルの処理である。

勘のいい方ならば、この方法には2つの問題があることにすぐに気づくだろう。ひとつはクライアント、あるいはプロキシサーバーでタイムアウトが発生しないか、という問題だが、これを解決するのは簡単である。タイムアウトの発生より短い周期で、ときどきサーバーがクライアントにダミーデータを戻せばいい。今回のサンプルは、この機能を搭載しており、調子よく動作している。

もうひとつの問題は、接続の切断後、すぐに再接続するとはいえ、その間に受信したデータが損失しないかということである。これを解決するにはサーバー側でデータをキューイング処理する必要がある。今回のサンプルは、切断から再接続までの間に、1件の発言があった場合は対応できるが、2件以上あった場合、データは失われる。キューイング処理を行なっていないからである。私は、これを回避するバージョンを作ろうとしたが、サンプルがHttpApplicationStateクラスを使っている関係で、エレガントに実装することができず、あきらめることにした。ただし、データをデータベースサーバーに保持するような方法をとれば、それほど無理なく実装はできる。

また、あまり学習することなく開発できるのが目的のはずだったのだが、結果的にはいくつかの技術を組み合わせて使わざるを得なければならなくなったのは、少々、残念である。特に発言受信用のプログラムと発言用のプログラムをひとつのexeファイルに収めるために、マルチスレッド化しなければならなかった。とはいえ、私も初めて.NET Windowsアプリケーションをマルチスレッド化したのだが、かなり簡単だった。ちなみに、送信用のプログラムと受信用のプログラムは動作的にはほとんど独立した別のプログラムなので、それらを2つのexeファイルに分離すれば、別にマルチスレッド化する必要はない。サーバー側も2つのaspxファイルで機能を実装しているが、これも発言受信用と送信用の2つのファイルである。

サンプルプログラムの構成とインストール

サンプルプログラムはひとつのexeファイル（クライアント：リスト1、リスト3）と2つのaspxファイル（サーバー：リスト2、リスト4）によって構成される。aspx