.NET Framework なにを使うか、どう使えるのか アイデアノート

第11回

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

P2Pアプリケーションの作成 ―その1―

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。

¥SAMPLE3
正常な終了処理を実装したチャットサーバー／クライアント

¥SAMPLE4
ネームサーバーとP2Pクライアント

P2Pアプリケーションの使用シナリオ

前回、P2Pアプリケーションを作ることを宣言し、マルチスレッドTCP/IPサーバーとクライアントのプログラミングを解説した。

どのようなP2Pアプリケーションを作るかのイメージはできたが、今号ではまだ完成していない。とはいえ、今回のサンプルは、ちゃんとP2Pアプリケーションとして動作する。もっとも、まだ、ピアツーピアで接続して、他のクライアントにキーワードを送信すると、それを受信した旨を返答するというだけの機能なので、何も利便性はない。前回、P2Pアプリケーションで分散データベースが作れないかと書いたが、一応、その方向で開発を進めている。

次号で完成するサンプルアプリケーションは次のように利用されるシナリオを想定している。

営業マンは自分のノートPCにこのP2Pアプリケーションをインストールする。アプケーションにはデータベース機能がついているので、社外でオフラインのときでも、彼が担当する顧客の情報をスタンドアロンのデータベース同様に管理することができる。

彼が会社に戻ってオンラインになると、同じアプリケーションが稼動している他の営業マンの顧客情報を参照できると同時に、他の営業マンも彼が持つ顧客情報を見ることができる。

本連載の第1回で「持ち運び可能データベース」を作成した。このシステムも同じような用途に使えるが、異なるのは帰社してオンラインにしたときに、自分が入力したデータをアップロードする必要がないということである。その一方で、あるユーザーがオフラインになると、その人が所有するデータを他の人は一切参照できなくなる。

あまり企業システムには向かない？

このように他者が管理するデータを利用できるかもしれないし、そうでないかもしれないという不確実性は、あまり企業の情報システムに向いていないかもしれない。また、管理者が集中的にバックアップを行なうことができないので、あるユーザーが自分の持つデータを喪失してしまうと、そのデータは永遠に失われることになる。

実のところ、P2Pアプリケーションというのは、ミニマムな仕様でも、どうしてもそこそこ複雑になってしまうため、本連載のようなページが少ない記事で扱うのは、少し無理がある。次号で完成するプログラムも最低限の実用性は持たせているが、細かい仕様上の問題点には目をつぶったままである。P2Pアプリケーションが複雑になってしまうのは、それぞれのクライアントがTCP/IPサーバーでもあるという事情による。

サンプルでは、各スレッドレベルの通信では同期処理を行なっているので、プログラムとしてはあまり複雑ではない。しかし、非同期処理を実現するためにマルチスレッドを採用しているため、その部分は少し理解しにくいだろう。本稿は技術解説記事なので積極的にマルチスレッドを採用したが、マルチスレッドを採用するべきかどうかには疑問も残る。というのは、接続する他のクライアントの数だけ新しいスレッドを立ち上げるのは、あまり、効率的な方法ではない。特にWindows 9x系のOSでは不安が残る。

Sample3 正常な終了処理ができるTCP/IPサーバー

Sample3（図1・2）は機能的には前回のSample2と同じだが、いくつかの障害を解決している。気づいた方もいると思うが、前回紹介したサンプルは、プログラムは動作するが、正常に終了することができない。開発環境で実行しているときはまだいいのだが、コンパイルしたサーバープログラムはウィンドウを閉じてもプロセスが残ってしまう。で、再度、起動しても残ったプロセスがポートを使用しているため、新しく起動したサーバープログラムはサービスを開始することができない。困ったことに、古いプロセスはそのままサービスを続けているため、プログラムは一見動いているように見えてしまうこともある。

前回は、とにかく簡単で読みやすいコードで、通信プログラムを紹介したかったので、あのような仕様になった。しかし、いくらサンプルとはいえ、これでは、まともなプログラムということはできない。また、クライアントとサーバー間でデータのやりとりをしていると、ときどき不要な改行が入ってしまうときがあった。そこで、今回はこれらの問題を解決したサンプルを用意した。

ところで、ここで紹介するプログラムの終了処理が標準的な方法かどうかは実はよくわからない。私流のひとつの解決方法だということを理解しておいていただきたい。

プログラムが終了できないのは新しいスレッドを起動するときに使用するサブルーチンが無限ループ内で停止しているからである。サーバー側におけるループは、クライアントからの接続を待つループとクライアントからのデータ受信を待つループである。また、クライアント側にもサーバーからのデータを待つループがある。しかし、問題はループだけではない。クライアントからの接続を待機するループは、接続したクライアントの数だけループ処理が実行される。つまり、プログラムはループによって停止しているわけではない。プログラムが停止しているのは、TcpListenerクラスのAcceptTcpClientメソッドの行である。つまり、プログラムを正常に終了させ、スレッドを閉じるには、この停止している行から次の行に移動させる必要がある。AcceptTcpClientメソッドが記述されている行が次の行に移るのは、クライアントからの接続が行なわれたときである。そこで、ここではサーバーに対して、自分自身で接続を行なっている。もちろん、それだけではループが回って次のクライアントの接続に備えてしまうので、無限ループから脱出するためのフラグを事前に立てておくことが必要である。

終了処理は、フォームのClosingイベントプロシージャで行なっている。接続のためのループを脱出する処理