.NET Frameworkで作るWindowsサーバー

作ればわかるアプリケーションの動作とメカニズム

第5回 非同期通信ソケットサーバーを作る

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

Windows Server

level

1 2 3 4 5

Technology Tools

- [x] Visual Basic
- [ ] Visual C#
- [ ] Visual C++
- [ ] SQL Server
- [ ] Oracle
- [ ] Access
- [ ] ASP.NET
- [ ] Other:

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、http://www.shoeisha.com/mag/windev/からダウンロード可能です。

非同期ソケットと同期ソケット

前号まで、自作のWebサーバーを作成した。そのTCP/IPサーバー部には「同期ソケット」をマルチスレッドで使用する方法を採用した。「同期ソケット」は非同期の通信ができないが、それを各接続ごとにマルチスレッドで使用することで、非同期の通信を実現していた。「同期ソケット」はプログラミングが単純だというメリットがあるが、非同期通信を実現するためには接続ごとに新しいスレッドを用意する必要がある。そして、スレッド数が増えるとタスクの切り替えのためのオーバーヘッドが発生する。サーバーに1000クライアントが接続し、1000のスレッドが起動しているところを想像してみてほしい。あまり、気持ちのいいものではない。

通常、非同期通信をするためには「非同期ソケット」を使ってプログラミングする方法が一般的である。この方法でプログラミングすれば、接続ごとにスレッドを起動する必要はない。ただし、プログラミングは複雑になる。また、同期通信でマルチスレッド化した場合と比較して、どのくらいリソースの消費が少ないか、またはオーバーヘッドが軽減されるかの明確な目安はない。それでも、サーバープログラムの作成方法を学習する上では必須の技術といえるだろう。また、次号では「非同期ソケット」のマルチスレッドの作成方法を紹介する。非同期通信をマルチスレッド化するのは、「同期ソケット」をマルチスレッド化する場合とは違って、通信の非同期化が目的ではない。各接続ごとにスレッドを作成するのではなく、たとえば10接続でひとつのスレッドというように1スレッドあたりの負荷を分散することに意味がある。これにより、マルチプロセッサコンピュータで使用する場合にハイパフォーマンスが期待できる。とはいえ、非同期通信をマルチスレッド化するプログラミングはさらに複雑である。だ

から、今回説明する簡単な非同期ソケットプログラミングの方法をしっかりと理解してもらいたい。

なお、今回のサンプルはサーバープログラムとクライアントプログラムにわかれるが、「非同期ソケット」を使用しているのはサーバー部だけである。非同期ソケットサーバープログラムだからといって、別に非同期ソケットクライアントとしか通信できないわけではないのである。

シンプルな非同期ソケットサーバー

送られた文字列をクライアントに送り返すサーバープログラム

サンプル1のサーバープログラム（図1）は、非同期ソケットを使ったシンプルなソケットサーバーである。フォームにはボタンがひとつあるだけである。このボタンをクリックすればサービスを開始する。図2は同サンプルのクライアント部である。サーバーを起動してからクライアントの［接続］ボタンをクリックすれば接続が完了する。クライアントプログラム下部のテキストボックスに何かを入力し［送信］ボタンをクリックすれば、入力した内容をサーバーに送信し、サーバーは同じ内容をクライアントに送り返す。これがこのサンプルプログラムの機能のすべてである。

図1：非同期ソケットサーバー

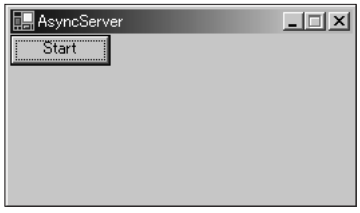

図2：非同期ソケットサーバーに接続する同期ソケットクライアント

プロシージャ間でデータの受け渡しをする「StateObject」クラス

リスト1は非同期ソケットサーバープログラムの全コードである。

まず、冒頭で「StateObject」クラスが宣言されていることに注意してほしい。このクラスはこの名前である必要もないし、このような構造である必要もないが、通常の通信アプリケーションでは、最低、次の3つの機能のメンバを持つクラスが必要になる。

①通信用のSocketを特定する
②1回の読み取りで受信したデータを格納するバッファ
③受信した総データを格納する

ただし、1回の読み込みで相手が送信した内容のすべてを受信できる場合、③のメンバはなくても問題ない。

サンプルプログラムでは、「StateObject」クラスにそれぞれ次のような要素を用意している。

①workSocket As Socket
②buffer As Byte
③sb As StringBuilder

これらのメンバはプログラム中で重要な役割を果たすので、まず構造を理解してほしい。

非同期ソケットプログラミングでは、ひとつのプロシージャで処理が完結しないため、プロシージャ間でデータの受け渡しをする必要がある。また、受け渡しは同時にひとつのオブジェクトしか行なえない。だからひとつのオブジェクトで複数のデータを渡すことのできる「StateObject」クラスのようなオブジェクトが必要になるのである。

また、複数のメンバを持つオブジェクトの受け渡しをしなくても、グローバルなスコープを持つオブジェクト配列に上記のような要素を格納し、プロシージャ間では識別子だけを受け渡すというような方法も考えられるだろう。