アドバンスド Windows プログラミング

Microsoftテクノロジーをめぐる冒険

第 7 回

レガシー機能のライブラリ

こだか かおる
KODAKA, Kaoru

はじめに

前回はインターネット関連のライブラリに注目してみましたが、今回はレガシー機能のライブラリについて見ていくことにします。レガシー機能とはいっても、それなりに必要になるものです。新しい機能だけではなく、古くからある、ある意味取り残されてしまった機能についても、.NET Framework 2.0では取り入れています。ちょっと不思議な感じはありますが、やはり需要に応えたものなのでしょう。

現時点では、「あれがほしい、これがほしい」というリクエストをあげるには遅すぎるでしょうが、.NET Framework 2.1、あるいは3.0で取り入れられるかもしれません。声を出さなければ、誰も気づかないものです。「この機能はないのか……」と落胆するのではなく、「これこれこういう理由で、○○の機能が必要だ！」と主張してみてはいかがでしょうか。

Microsoftも積極的に開発者、ユーザーの声を取り入れようとしています。たとえば、Product Feedback Centerでは、バグレポートだけではなく提案（Suggestion）をすることも可能です。よりよい製品になるように、いろいろと努力をしているというわけです。

Level

1 2 3 4 5

Technology Tools

- ◯ Visual Basic
- ✓ Visual C#
- ◯ Visual C++
- ◯ SQL Server
- ◯ Oracle
- ◯ Access
- ◯ ASP.NET
- ✓ Other:
  - ↓
  - Visual Studio 2005ベータ2
  - SQL Server 2005 Express Edition April CTP

プロセス間通信

アプリケーションとアプリケーションの間でデータをやり取りする、いわゆる「プロセス間通信（Interprocess Communication）」は、ときとして必要になる機能のひとつです。ウィンドウメッセージ、DDE、メモリマップドファイル、メールスロット、名前付きパイプ、ソケットなど、いろいろな方法が用意されていますが、いわゆるSDKプログラミングでは簡単に利用できるためか、ウィンドウメッセージを用いるのがもっとも一般的な方法でした。

もちろん.NET Frameworkの時代になっても、プロセス間通信の需要はなくなりません。ところが、.NET Frameworkに用意されている通信手段は、ソケット、あるいは、リモーティングのみだったのです。どちらも簡単に利用できる方法ではありません。仕方がないので、多少大掛かりですがソケット、リモーティングを使ったり、あるいはP/Invokeを利用してウィンドウメッセ

ージを使ったり、といった実装方法が選択されているようです。

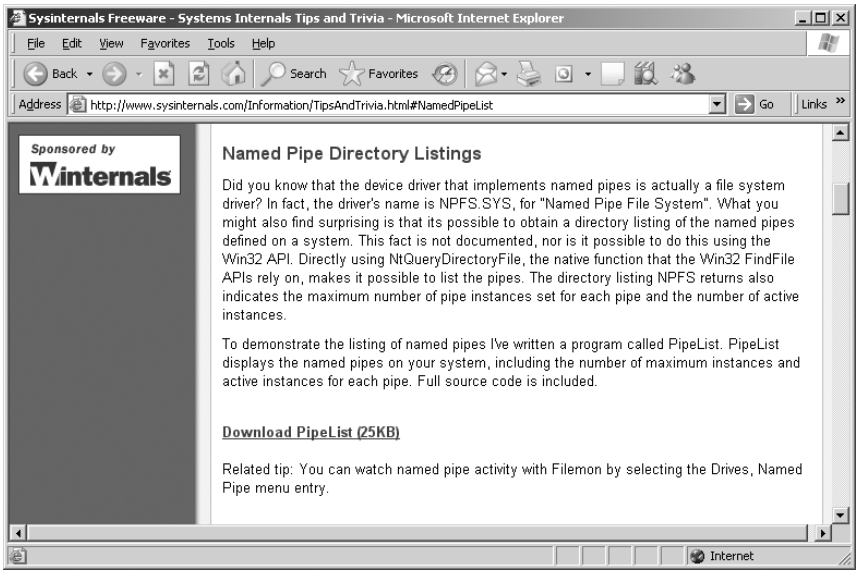
◎.NET Framework 2.0では？

.NET Framework 2.0にはプロセス間通信用として、リモーティングの一部にIpcという名前空間、クラス群が用意されています。以前にもこのリモーティングを試したことがあるのですが、そのときはどうもうまくいきませんでした。というのも、参照設定（System.Runtime.Remoting：System.Runtime.Remoting.dll）はできているものの、名前空間の指定を行なっても、リストに出てこなかったのです。当然、クラスもインテリセンスに現われず、無視して直接クラスを書いてもダメでした。これはおそらく、アセンブリの登録などに何らかの問題があったのでしょう。原因究明に時間がかかりそうだったため、オペレーティングシステムからクリーンインストールをし直しました。

今回は気分を一新し、このリモーティングについて見ていきたいと思います。いきなりですが、リモーティングはそれほどプロセス間通信に向いたものではありません。プロセス間通信といえば、送信先を指定してデータを送信、受け取り側では送られたデータをもらって、何らかの処理をする、というやり取りが一般的です。しかし、リモーティングの主目的は、「リモートのオブジェクトをクライアントからの指示により生成する」というものです。簡単なデータを送受信するという形での使い方はあまり想定されていないように思われます。

とはいえ、せっかく用意されているクラスですので、それを使わないのももったいない気がします。そこで、リモーティングによるプロセス間通信を試してみましょう。

図1：Named Pipe Directory Listings



プロセス間通信用に用意されているIpcリモーティングですが、サンプルプログラムを見ると、ホスト名とポートを指定しています。このことから、当初はTCP/IPを利用したものになっているのかと思っていました。しかし、TCP/IPベースのやり取りでは必ず反応する、ノートンインターネットセキュリティが反応しません[注1]。また、netstatコマンドで待ち受けポートを確認してみても、一覧に現われないのです。

また、サンプルで指定している「ホスト名:ポート」を、適当な文字列「Ipc SamplePort」などに変えてみたところ、特に何の問題もなく動いてしまいました。

これらを踏まえると、どうやら通信はTCP/IPベースではないようです。では、どのようなテクノロジーで実現されているのでしょうか？　いろいろな調べ方があるわけですが、レガシーなプロセス間通信の中で、名前を指定するといえば「名前付きパイプ」です。まずは、そのあたりから調べてみることにしましょう。

何か便利なツールがないかと思って、Webサイトをさまよっていたところ、SysInternalsで名前付きパイプの一覧を表示する「PipeList」というアプリケーションを見つけました（図1）。

http://www.sysinternals.com/Information/TipsAndTrivia.html#NamedPipeList

このアプリケーションを利用して、サンプルプログラムの実行時に確認をしてみたのが図2になります。一番下に、

注1）ノートンインターネットセキュリティには、アプリケーションがTCP/IPベースの通信を行なう際、通信を始めてよいかどうか確認する機能があります。トロイの木馬やワームなど、悪意あるアプリケーションの存在を想定しています。