.NET Framework アイデアノート

なにを使うか、どう使えるのか

第3回

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

OLE DB用、SQL Server用を簡単に切り替え可能なADO.NETラッパークラス―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。

CLASSLIBRARY1.SLN
独自クラスのソリューションファイル

SAMPLE1～4.ASPX
独自クラスの利用例サンプル

DB.INC
全サンプルが参照するインクルードファイル

うれしいサードパーティ製の開発環境の登場

.NET Frameworkをめぐるプログラミングで、現在、最大のトピックは本誌2003年11月号でも紹介されているように、Visual Studio .NET以外の有力な開発環境が現われたことだろう。特にBorland C#Builderは、付属のコンポーネントが多いことが魅力的だ。また、Personal版は無料であることが魅力的である。

私はVisual Studio .NET 2003を持っていないので、これを使えば.NET Framework 1.1のプログラミングをRAD環境で使えることになる。一方でVisual Studio .NETがマルチランゲージに対応しているのは、やはり魅力的である。私はVisual Basic .NETでプログラムを書かないといけないことも多いので、C#にしか対応していないC#Builderだけを使うという選択肢はない。だからVisual Studio .NET、もしくはSharpDevelopを使うか、それらとC#Builderを組み合わせることになるだろう。基本的に.NET Frameworkプログラミングは無償で配布される.NET Framework SDKに依存しているので、環境の提供者であるMicrosoft製品のアドバンテージは、それほどないように思える。Microsoftがコンパイラを含む.NET Framework SDKを無償で提供していることは、十分に高く評価すべきだ。それにより、他者は大きな開発コストをかけることなく、独自ツールを我々に提供することができるのである。

プログラミングスタイルを決めるのはライブラリ

16bitのWindowsアプリケーション開発ツールが世に出始めたとき、Microsoftの開発ツールはメジャーな存在ではあったが、今日のように圧倒的な存在ではなかった。い

くつものツールが一般的に利用されている今日のJavaプログラミング環境の状況に似ているのかもしれない。.NET Frameworkプログラミングにおいても、再び、そのような時代がくることを願うばかりである。ただ、.NET Frameworkを使用する以上、ライブラリも言語もコンパイラも共通なので、開発中の使い勝手を除けば、大きな特色を出しにくいのが残念である。それを乗り越えるには、新しいクラスが追加されればいいわけで、それを実践しているのがC#Builderに付属するComponent One社のコンポーネントだろう。

従来のWindowsアプリケーションツールも結局は、Windows APIベースで動作するという意味では、どのツールで開発したプログラムも共通だった。ただ、それをラップするライブラリの違いによってプログラミングスタイルを変えることができ、それが開発ツールの特色となっていた。だから、どの.NET Frameworkアプリケーションも.NET Frameworkベースで動作することは同じでも、ライブラリの違いによってプログラミングモデルを変更できるはずである。だから、今後の開発ツールに追加されるクラスは、単に新機能を追加するためのクラスばかりではなく、プログラミングスタイルを変化させるようなタイプのクラスの追加も期待したい。Microsoftが提供するクラスは資料も不備だし、決して使いやすいものではないので、そこに開発ツールの特色を出すチャンスはあるはずである。

繰り返すが、プログラミングスタイルの多くは、言語で決まるのでもIDEで決まるのでもなく、ライブラリで決まるのである。.NET Frameworkが提供するクラスの多くは、より多くの機能を羅網することに力が注がれており、必ずしも快適で洗練されたプログラミングのために提供されているわけではない。だから、MFCに対するOWLのようなライブラリが、やはり、それこそBorlandから発売されないかと私は期待してしまうのである。ちなみに以前のVisual Basicは、他を犠牲にしても快適なプログラミングを目指す傾向が強かったと思う。

ADO.NETをラップした独自クラス

今回はADO.NETをラップした独自クラス（UDataAccessクラスとUDataResultクラス）の利用法を紹介する。このクラスはプログラミングスタイルを提供するクラスであって、新機能を提供するものではない。このクラスの最大のメリットは、OLE DB用のADO.NETクラスとSQL Server用のデータアクセスクラスを簡単に切り替えることができることにある。

パッケージソフトのように、SQL Serverがある場合には高速なSQL Server用のクラスを使い、ない場合にはOLE DB用のクラスを使いたい場合がある。これらのクラスは引数で切り替えられるわけでなくオブジェクトの宣言で使い分けないといけないので、If文で分岐させると冗長になってしまう。

UDataAccessクラスとUDataResultクラスは接続文字列の内容によって、これら2つのオブジェクトを切り替える。つまり、接続文字列に“provider”という文字列が入っていたら、それはOLE DBでアクセスするデータベースだと判断する。もし、SQL Serverを使うケースでデータベースサーバー名やアクセスするデータベース名にこの文字列が含まれている場合や明示的にSQL Server用のオブジェクトを使用したい場合は、UDataAccessオブジェクトのDataTypeプロパティに“SQL”と指定すればいい。

このクラスを使えばプログラム記述が簡単

このクラスのもうひとつの長所は、UDataAccessオブジェクトとUDataResultオブジェクトの2つだけを宣言すればデータベースへのアクセスが可能になることである。

ADO.NETになって、データベースへアクセスするための通常の記述はかなり複雑になった。DataReaderオブジェクトを使ってデータアクセスする場合は、Connectionオブジェクト、Commandオブジェクト、DataReaderオブジェクトを宣言しないといけないし、DataSetオブ