新　連　載

.NET Framework アイデアノート

なにを使うか、どう使えるのか

第1回

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

ADO.NETのマルチテーブル機能 ～「持ち運び可能なデータベース」を実現する

Technology Tools

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

Level

★★½☆☆

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。

・WINDOWSAPPLICATION1.SLN
今回のサンプルプログラムのソリューションファイル

言語、プログラミング、創造性

.NET Frameworkが巨大な技術体系であることに異論をはさむ人はいないだろう。また、それに付随するさまざまなテクノロジーを加えれば、そのマスはさらに大きくなる。フレームワークとして提供されるさまざまな機能には、それぞれの目的がある。われわれ開発者は、その機能を理解して自分のやりたいことに利用できるかどうかを検討し、適しているならば採用する。ことの是非はともかく、これが今日のプログラミングスタイルであることは事実だろう。

.NETプログラミングを行なううえで、私たちが提供される機能の内容や目的を把握する最初の手がかりは、Microsoftから提供されるドキュメントであることが多い。たとえば、クラスの作成者は何かの目的を実行するためにクラスのメソッドを実装し、それに則してドキュメントを作成する。アプリケーションの開発者は、そのドキュメントを頼りにプログラムをどのように実装するかのイメージを固めるわけだが、そこにはいくつかの問題と制約がある。

まず、問題としては、ドキュメントの不備によって機能の目的がはっきりしない場合がある。これは.NETテクノロジーを扱うほとんどの開発者にとって現時点で、かなり、大きな問題となっていると想像している。.NET出現以降のMicrosoftのドキュメントを読んでいると、私にはどうも抽象的な表現が多く説明不足のように思える。

とはいえ、抽象的な表現には、イメージを制約しないというメリットもある。先に書いた制約とは、まさにこのことである。クラスによって提供された機能は、必ずしも作成者が意図した目的通りに使われる必要はない。ドキュメントが具体

的に書かれるほど、それを読んだものは作成者の意図した目的に呪縛される。言語の各要素を本来の意図より拡張して利用することを文学とするならば、要素の意図に束縛されるプログラミングは文学的とはいえない。言語表現による行為の一種であるプログラミングは創造的であっていいはずである。

この連載の目的

この連載では、.NET Frameworkとその周辺技術が提供する機能の一部にスポットを当て、その目的について検討し、結果的にその機能を使ってどのようなことができるかを私的に解釈したものを具体例として提供する。扱うのはWebアプリケーション、Windowsアプリケーションを問わない。

Microsoftの意図と合致するかはともかくひとつの解釈として読んでもらい、何かしらのアイデアが触発されればこの連載の目的は達成できたことになる。たとえば、今回はADO.NETのDataSetクラスのマルチテーブル機能を検証し、「持ち運び可能なデータベース」として実装した具体例を提供する。

持ち運び可能なデータベースとは？

持ち運び可能なデータベースとは何かを簡単に説明しておこう。普通、持ち運び可能といった場合、「ポータブル」と訳されると思う。しかし「ポータブル」をコンピュータ用語として日本語に訳す場合は「移植可能」である。今回提供するサンプルは、そういう意味ではなく、たとえばノートブックパソコンに入れて持ち運べるという意味でまさに「持ち運び可能」なのである。具体的なオペレーション例を下記に示す。

①ユーザーはデータベースサーバーにアクセスして、ある程度まとまったデータをダウンロードする。
②ダウンロードしたデータを編集してファイルに保存し、コンピュータ（あるいはアプリケーション）をシャットダウンする。ここでユーザーはそのコンピュータを持ってどこかに移動するかもしれない。
③ユーザーは再度、コンピュータを起動し、保存してあるファイルをロードし、編集作業を続行する。
④編集の終了後、データベースサーバーに接続して、編集内容をデータベースに反映する。

これが私のいう「持ち運び可能なデータベース」のオペレーション概要である。データベース用語でいえば、“データレプリケーション”が一番近いだろう。しかし、もちろん、クライアントコンピュータにはADO.NET以外にデータベースエンジンをインストールする必要はない。なぜなら、マルチテーブルが可能になった時点で、ADO.NETのDataSetクラス自体がインメモリでリレーショナルなデータベースエンジンといえるからである。

ADO.NETのマルチテーブル機能の目的

ADO.NETがマルチテーブルをサポートしたことを聞いたとき、あまりよく考えなかったせいもあって、私はその意図がよくわからなかった。Microsoftは非接続型のモデルを実現するために、この機能を実装したのだという。つまり、「非接続モデルの実現」が、Microsoftの想定するこの機能の目的なのである。

先に結論をいっておくと、Microsoftが提供した機能だけでは上記の「持ち運び可能なデータベース」は実現できない。Microsoftが想定しているのは、あくまでダイヤルアップ接続で、一度、接続したデータベースと切断して編集を行ない、（アプリケーションを閉じることなく）再度、データベースに接続して更新することができるだけである。つまり、一度、アプリケーションをシャットダウンしてしまうとデータベースの更新はできない。これでは電話代が節約できることと、一度、ダウンロードしておけば低速回線でも、それ以降データを参照するのが高速であることくらいしかメリットがない。常時接続環境が急速に普及している現状で、これはたいしたメ