.NET Framework なにを使うか、どう使えるのか アイデアノート

第4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2―

～DataReaderとDataSetのプログラミングスタイルの差を吸収する

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。

¥UDA2¥UDALIB.SLN
独自クラスのソリューションファイル

¥UDA2¥SAMPLE1～3.ASPX
独自クラスの利用例サンプル

¥DIFFGRAM
持ち運び可能なデータベース改訂版

長所と短所

先月紹介したデータベースアクセスのためのラッパークラスUData Accessクラスを強化したものを今回は解説する。先月号を読んでいない方でも、今月号を読めば、使い方はほとんどわかるはずである。新しいUDataAccessクラスは先月号で発表したものの上位互換なので、先月号で解説したプログラム記述方法でも使用できるが、できれば今回解説する方法でプログラムを作ってほしい。

実用に耐える優れたものになったので、ぜひ、読者の方にも利用していただきたい。私自身、今後、ADO.NETを使ったデータベースプログラムの開発には、すべてこのクラスを利用する予定である。

ADO.NETを直接利用した場合と比較したメリットとしては次のようなことがあげられる。

メリット1▶データベースアクセス処理の記述が簡単

メリット2▶OLE DB用のクラスとSQL Server用のクラスを1行の修正で切り替えられる

メリット3▶DataReaderクラスとDataSetクラスを1行の修正で切り替えられる

このような長所がある一方で柔軟性はADO.NETでプログラムを直接記述した場合と変わらない。

対するデメリットは、ひとつの接続でひとつのDataSetオブジェクトしか利用できない点である。あと、もちろんVisual Studio .NETのコード生成機能は、UDataAccessクラスに対応するコードを生成してくれない。

以下、これらの長所と短所について、詳しく説明する。

最小4行のコードでDBからデータを取得

先月号ではUDataAccessクラスとUDataResultクラスの2つを提供した。最新版でも互換性のためにUDataResultクラスは提供されるが、もはや使用する必要はない。UDataAccessクラスだけを宣言すれば、結果セットまで作成することができる。結果として後掲のサンプル1（リスト1）のように、さらに簡単なコード記述でデータベースにアクセスできるようになった。クラスの宣言もひとつだけでよいし、変数の宣言と変数に値を代入する部分を除けば、1行でデータベースに接続でき、さらに1行でクエリを実行でき、さらに1行で結果セットの内容を取得できる。

つまり、もっとも短い記述方法ならば、次の4行でデータベースから取得したデータを1行表示することができる。

```
Dim UDA As New UDataAccess
UDA.Open("Data Source=localhost;Initial Catalog=pubs")
UDA.GetResultSet("DataReader", _
                 "SELECT au_id,au_lname FROM authors")
Response.Write( _
 UDA.DataResults("DataReader").Item("au_lname"))
```

もちろんOLE DB用とSQL Server用のクラスの差も吸収

今回もサンプルはASP.NET用のものだけを用意したが、もちろんWindowsアプリケーションでも利用することができる。

UDataAccessクラスのメリットは、OLE DB用のADO.NETのクラスとSQL Server用のクラスを簡単に切り替えられる点にあったことを覚えている読者もいるだろう。新しいバージョンでは、さらにDataReaderクラスとDataSetクラスも簡単に切り替えられるようになった。上記の4行のプログラムコードで“DataReader”という文字列が指定されている部分に、それ以外の任意の文字列を指定すれば、データベースへのアクセスはDataReaderオブジェクトではなく、DataSetオブジェクトを使用して行なわれる。つまり、この文字列をサンプル1（リスト1）のプログラムのように先頭で変数に入れておけば、1行変更するだけで、DataReaderクラスとDataSetクラスを切り替えて使用することができる。

DataReaderクラスとDataSetクラスのプログラム記述方法の極端な違いを知っている方ならば、このメリットが大きいことがわかるはずだ。SQL Serverからの結果セットの行数が多く、かつ、前方の結果しか使用しない場合、DataReaderオブジェクトはDataSetクラスよりも大幅に速く処理を終了する。だから、できる限り、DataReaderクラスを使いたいのだが、開発している最中にDataReaderクラスの機能では目的を達成できないことが判明することもあるだろう。たとえば、結果セットを2度走査したいような場合、カーソルを前方に移動できないDataReaderオブジェクトでは再度クエリを実行しなければならない。よくあるのは結果セットの行数を知りたい場合だが、その場合、DataReaderオブジェクトのパフォーマンス上のメリットはなくなる上、行数を取得するためのコードを記述しなければならない。しかし、だからといってDataSetクラスを使ったプログラムに書き換えるのはかなりの修正が必要になる。

SQL Server用のクラスとOLE DB用のクラスのプログラミングの違いは宣言するクラスが違うだけなので機械的に変更できるが、DataReaderクラスとDataSetクラスはプログラミング構造自体が違うため、変更作業にはかなり神経をとがらせる必要がある。UDataAccessクラスを使えば、プログラミングの初期の段階で重大な決断が迫られなくなる分、ストレスは軽くなるはずだ。それ以前にDataReaderクラスとDataSetクラスの複雑な構文を2つも覚えなくてよいというメリットも大きい。

先月号でも書いたが、内部的に使用されるADO.NETのクラスは、すべてプロパティによって公開されているので、ADO.NETで直接プログラミングする場合と比較して柔軟性が劣るということはない。つまり、ADO.NETのスキルはすべて利用できる。簡単な部分はADO.NETよりもずっと簡単な記述とスキルでプログラミングでき、難しい部分はADO.NETと同じようにプログラミングできるというのがUDataAccessクラスの特徴である。クラス実装の内容の解説は次号で行なうが、UData