Visual Basic .NETのツボ

第25回 ADO.NETを利用したデータベース処理 ―その5―

西田 雅昭
NISHIDA, Masaaki

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¥TUBOディレクトリに収録しています。

¥ADOCODE3
今回作成したサンプル

¥DATA
サンプルデータベース（Shimei.mdb）

*）サンプルプログラムは、データファイルを「E:¥dotNETMagazine¥VB21¥Data¥」ディレクトリに配置しているという前提で作成しています。サンプルの実行時には、上記のディレクトリにデータファイルを配置するか、以下の2つを自分の環境に合わせて修正する必要があります。

・「btnConnect_Click」プロシージャの接続文字列
・「OleDbConnection」コントロール（OleDbConnection1）の「ConnectionString」プロパティ

前回「ADOTest3」プロジェクトを作成しましたが、[Adapter] [Select] [DataSet] [Fill2]という4つのボタンのコードについては詳しく説明しませんでした。そこで今回は、これらのコードを解説しつつ、「データアダプタ」と「データコマンド」について、さらに理解を深めることにしましょう。

データアダプタ [Adapter] ボタンのコード

まずは、「データアダプタ」の説明から始めます。データアダプタは、後で詳しく説明する「データセット」へデータソースを格納したり、データセット上で行なわれたデータの更新をデータソースに返したりする、いわば“ブリッジ”のような働きをするオブジェクトです。

そのため、データアダプタは、一連のデータコマンドやデータ接続に関する機能を持っています。

「ADOTest3」プロジェクトを起動して、[Adapter] ボタンをダブルクリックしてください。イベントハンドラのコードを見ることができます（リスト1）。

最初の5行（リスト1-①）は、4つの「OleDbCommand」オブジェクトと、ひとつの「OleDbDataAdapter」オブジェクトのインスタンスを作成しているだけです。

次のWith構文（リスト1-②）は、この4つのコマンドをデータアダプタが利用することを宣言しています。

たとえば「OleDbDataAdapter.DeleteCommand」は、データセットからレコードを削除するためのSQLステートメントを設定するプロパティです。これ以外の3つも名前から想像できますね。このプロジェクトでは、まだ「SelectCommand」しか使っていないのですが、後々のために4つともきちんと記述しておきました。

◉DataTableMapping CollectionとAddRange

さて、見るだけで、なんともいやになるのが最後の1行です。あまりに長いので、リスト1ではわかりやすいように改行し、字下げを付けておきました。

```
modaShimei.TableMappings.AddRange _
 (New System.Data.Common.DataTableMapping() _
  {New System.Data.Common.DataTableMapping _
     (略)
     } _
   ) _
  } _
 )
```

まず「TableMappings」は、「DataTableMappingCollection」コレクションを取得するプロパティです。では「DataTableMappingCollection」コレクションとは何かというと、「DataTableMapping」オブジェクト（データソースのテーブルとデータソース内のテーブルとの間の対応関係）」の集合です。

「AddRange」に関しては、少し説明が必要です。多くの場合、.NET Frameworkのコレクションは、コレクションにオブジェクトを追加するために、「Add」と「AddRange」という2つのメソッドを持ちます。

「AddRange」は配列の要素をコレクションの末尾にコピーするメソッドです。「配列をコピーする」とは、どういうことでしょうか。

通常、

```
Add xxx
Add yyy
Add zzz
```

のように記述するところを、「AddRange」メソッドを使うと、

```
Dim hairetsu() {xxx, yyy, zzz}
Addrange(hauretsu)
```

のように記述できるということです。

また、

```
AddRange(New hairetu() {xxx, yyy, zzz})
```

のような記述も可能です。これは、「{xxx, yyy, zzz}という要素を持つ配列を新たに作成して引数として使う」という意味になります。

参考のために「DataTableMapping Collection.Addrange」メソッドの書式を載せておきましょう（表1）。

先ほどのコードの場合、「DataTable MappingCollection」コレクションに追加するオブジェクトは、2行目の「DataTableMapping()」という配列ということになります。

3行目の最初の“{”は、この配列の要素の一覧の始まりを表わします。ただし、ここでは「DataTableMapping」オブジェクトはひとつしかありません。

◉オーバーロード

ここで「DataTableMapping」のコンストラクタの解説に移る前に、「New」について詳しく説明しておきましょう。

「New」キーワードを使って、クラスのインスタンスを作成することは、

表1：DataTableMappingCollection.Addrangeメソッドの構文

Public Sub AddRange(ByVal values() As DataTableMapping)		
機能	指定したDataTableMappingの配列を「DataTableMappingCollection」コレクションの末尾にコピーする	
パラメータ	values	コレクションに追加するDataTableMappingオブジェクトの配列

リスト1：［Adapter］ボタンのイベントハンドラに記述したコード

```
mscmdShimei = New OleDb.OleDbCommand
micmdShimei = New OleDb.OleDbCommand
mucmdShimei = New OleDb.OleDbCommand
mdcmdShimei = New OleDb.OleDbCommand
modaShimei = New OleDb.OleDbDataAdapter

With modaShimei
   .DeleteCommand = mdcmdShimei
   .InsertCommand = micmdShimei
   .SelectCommand = mscmdShimei
   .UpdateCommand = mucmdShimei
End With

modaShimei.TableMappings.AddRange _
  (New System.Data.Common.DataTableMapping() _
  {New System.Data.Common.DataTableMapping _
    ("Table", _
    "tbShimei", _
    New System.Data.Common.DataColumnMapping() _
      {New System.Data.Common.DataColumnMapping( _
        "Code", "clCode"), _
      New System.Data.Common.DataColumnMapping( _
        "Name", "clName"), _
      New System.Data.Common.DataColumnMapping( _
        "FName", "clFName") _
      } _
    ) _
  } _
 )
```