Visual Basic .NETのツボ

第30回 実用的なシステムにするための工夫

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。

¥OLD
前回作成したサンプルプログラム

¥ADOCODE4
サンプルプログラム完成版

¥DATA
サンプルデータベース（Shimei.mdb）

*）今回の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。サンプルを実行するには、上記のディレクトリにデータファイルを配置するか、「mprc ConnectMake」プロシージャの接続文字列を自分の環境に合わせて修正する必要があります。

本連載では、ADO.NETを使ったデータベース処理に関して、いろいろな実験を繰り返してきました。今回は、まとめとして、前回作成したプログラムに、実用的な機能を追加してシステムを完成することにします。

プログラムを動かしてみよう

さっそく前回作成したプロジェクト「ADOTest4」を起動することにしましょう。「ADOTest4.sln」をダブルクリックします。

そして、プロジェクトを実行すると、コネクションをオープンしたことを示すメッセージボックス（図1）が開きますから、［OK］ボタンを選択してください。

続いて、コネクションをクローズしたことを表わすメッセージボックス（図2）が開きますから、やはり［OK］ボタンを選択すると、「35行のデータを読み込みました」というメッセージ（図3）が出ます。［OK］ボタンを選択すると「ADOの実験」というウィンドウ（図4）が開きます。

最初のメッセージと2番目のメッセージボックスは、「mevhConnectionStateChange」イベントハンドラの次のコードで表示しているのです。

```
MessageBox.Show("以前の状態は " _
  & e.OriginalState.ToString _
  & ControlChars.CrLf & "現在の状態は " _
```

図1：コネクションのオープンを示すメッセージ

図2：コネクションのクローズを示すメッセージ

図3：読み込んだデータ数を示すメッセージ

図4：「ADOの実験」プログラムの起動画面

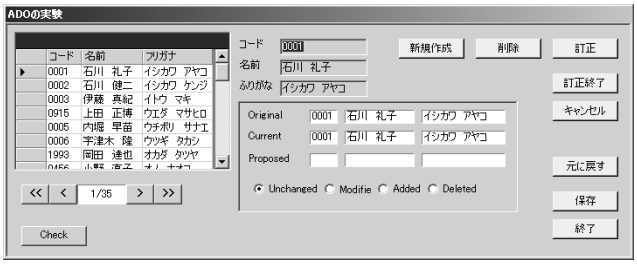
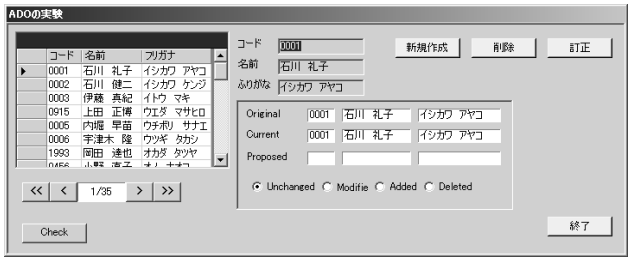


図5：起動画面で表示するボタンを制限




```
 & e.CurrentState.ToString)
End Sub
```

このイベントは、「Fill」メソッドを実行する前と後で発生します。これは、ADO.NETでは「Fill」メソッドを実行する前に、自動的に接続を開き、実行後は接続を閉じてしまう、ということを確認するために記述したコードです。

最後のメッセージボックスは、「mprcFill」プロシージャで、「Fill」メソッドが働いた後で記述しています。

```
MessageBox.Show(mintRecordCount.ToString & _
 " 行のデータを読み込みました")
```

実際のプロジェクトで、煩雑に感じる場合には、この3つのメッセージボックスの表示処理は消去しても構いません。

起動時に表示するボタンは？

起動画面で、緑色になっているのが実際に使用するボタンなのですが、すぐに使用するのは［新規作成］［削除］［訂正］［終了］の4つのボタンです。そこで、ユーザーが混乱しないように、これ以外のボタンは消してしまいましょう。

この処理は、プロシージャにしておけば、あとで使い回しができますね。

```
Private Sub mprcButtonBrowse()
   btnAddNew.Visible = True
   btnDelete.Visible = True
   btnEdit.Visible = True
   btnQuit.Visible = True

   btnEditEnd.Visible = False
   btnCancel.Visible = False
   btnOriginal.Visible = False
   btnSave.Visible = False
End Sub
```

プロシージャができたら、「mprcTextBoxLock」プロシージャの最後に、

```
mprcButtonBrowse()
```

という1行を追加します。これで起動画面では、ボタンが4つだけになりました(図5)。

操作の説明を表示しよう

私は、カスタムソフトは操作説明書などなくても使えるようにするべきだという考えを持っています。これを実現するには、ユーザーがそれぞれの時点で何をすればよいかを画面で指示する必要があります。

まずラベルをひとつフォームに配置しましょう。このラベルのプロパティは、次のように指定します（図6）。

プロパティ		値
(Name)		lblMessage
BackColor		White
BorderStyle		Fixed3D
Font	Name	MS明朝
	Size	9pt
	Bold	True
ForeColor		Navy

ここに表示する内容は、文字列定数にしておくのがよいでしょう。

```
Private Const mcstrMSGBrowse As String = _
 "レコードを移動するには左下の" & _
 "矢印キーを打ってください。" _
 & ControlChars.CrLf _
 & "データ処理の際には、機能を" & _
 "ボタンで選択してください"
```

先ほどと同じように、「mprcTextBox

図6：フォームにラベルを追加