使いやすいコントロールを手に入れるためのノウハウ

Visual Basic .NETで作るカスタムコンポーネント

第5回 データ対応カスタムコントロールの作成

ウェブデ・ネット有限会社
黒川 洋二 *KUROKAWA, Youji*
http://www.webde.net/

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¥NETARCH_CTLディレクトリに収録しています。

¥DATA
テストデータ（Sample.mdb）

¥DATANAVI
今回解説したカスタムコントロール／動作確認用プロジェクトを含むソリューションファイル

¥DRAWCONTROLS
カスタムコントロール「DrawButton」のプロジェクトファイル（前号で解説）

はじめに

今回は、カスタムコントロールにデータ機能を付加させてデータソースをプロパティページで設定できるようにし、CurrencyManagerクラスを使用したVB6.0時代のレコードを移動するデータコントロール風のカスタムコントロールを作成します。

具体的には、ユーザーコントロールに［先頭］［前へ］［次へ］［最後］といったデータ行を移動する各ボタンと「DataSource」プロパティを追加し、データ移動用のボタンを押下した際に「DataSource」プロパティで指定したデータオブジェクトの行を移動できるようにする方法について解説します。

データソースの概念

既存のWindowsコントロールのデータソースは、従来のデータベースに限らず任意のデータソースにバインドすることができます。たとえば、独自にカスタムデータソースを作成するのであれば、データソースの仕様として次に挙げるインターフェイスを実装する必要があります。

- System.ComponentModel.IBindingList
- System.ComponentModel.IEditableObject

IBindingListインターフェイスは、ListChangedイベントを発行して、リストの項目またはリスト自体の変更をプログラムに通知します。

IEditableObjectインターフェイスは、DataRowViewクラスに実装されています。このインターフェイスはBeginEdit、EndEdit、CancelEditの各メソッドを公開します。

データのバインディングを行なうには「System.Collections.IList」「System.ComponentModel.IComponent」といったインターフェイスを実装、または実装しているコントロールを継承している必要があります。

本稿で取り扱うデータソースの内容

はDataSet内の単一データメンバ（データテーブル）を指定して使用します。

バインディング

.NET Frameworkで定義されているWindowsフォーム上のデータのバインディングは、さまざまなオブジェクトの組み合わせにより機能しています。そのためバインディングはいろいろと複雑です。ここでは、バインディングを行なうための主要なクラス、BindingContextとCurrencyManagerについて簡単にまとめてみます。

BindingContextオブジェクト

「Control」クラスを継承するクラスには、Bindingの最上位にあるBindingContextオブジェクト（System.Windows.Forms.BindingContext）が実装されています。BindingContextオブジェクトは、BindingManagerBaseオブジェクトを管理します。BindingManagerBaseオブジェクトは、抽象クラスなので直接参照できません。BindingContextオブジェクトは、PropertyManagerクラスとCurrencyManagerクラス（これらはSystem.Windows.Forms名前空間に属します）のインスタンスを管理しています。

データソースが単一のデータを返す場合は、BindingManagerBaseのインスタンスとしてPropertyManagerオブジェクトが生成されます。データソースがオブジェクトのリストを返す場合は、インスタンスとしてCurrencyManagerオブジェクトが生成されます。ADO.NETでは常にインスタンスとしてCurrencyManagerオブジェクトが生成されます。

これらのことからBindingContextオブジェクトは、Controlクラスから派生したコントロールオブジェクト（Formやカスタムユーザーコントロールなど）にひとつだけ存在し、かつCurrencyManagerオブジェクトを取得できることがわかります。

リスト1に使用例を挙げてみましょう。ここで注目していただきたいのは以下のコード（リスト1の①）です。

```
Tbl = DataSet1.Tables(0)
CurrentData = Me.BindingContext(Tbl)
MsgBox("RowNo:" + CurrentData.Position.ToString)
```

ここでは「Me」（Formのインスタンス）から「BindingContext」にアクセスし、引数にDataSet1（データセットのインスタンス）が保有しているテーブルを設定して、DataSet1のTables(0)（DataTableオブジェクト）のCurrencyManagerオブジェクトを取得しています。

また、このメソッドの実行結果を図1に示します。のちほど詳しく説明しますが、DataGrid上で選択したデータをメッセージとして出力しています。

リスト1：BindingContextの使用例

```
Private Sub btnGetBindingContext_Click( _
  ByVal sender As System.Object, ByVal e As System.EventArgs) _
  Handles btnGetBindingContext.Click

    Dim CurrentData As CurrencyManager
    Dim Tbl As New DataTable
    Dim s As String

    Tbl = DataSet1.Tables(0)                                    ┐
    CurrentData = Me.BindingContext(Tbl)                        ├─①
    MsgBox("RowNo:" + CurrentData.Position.ToString)            ┘

    For Each Itm As Object In CType( _                          ┐
      CurrentData.Current, DataRowView).Row.ItemArray           ├─②
        s += CStr(Itm) + vbCrLf                                 ┘
    Next
    MsgBox(s)
End Sub
```

図1：BindingContextの使用例の実行結果