

Visual Studio .NETで 究めるコンポーネント開発

第5回

コントロールの設定値を永続化する

株式会社コムラッド
辻慶 TSUJI, Kei
<http://www.comrade.co.jp/>

Technology Tools

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

Level



Samples

・本稿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¥NETARCH03ディレクトリに収録しています。

¥CONSTANTPERSISTEDPROPERTYSAMPLE
パターン1のサンプル

¥PROPERTYPERSISTEDBYCONSTRUCTORSAMPLE
パターン2のサンプル

¥PROPERTYPERSISTEDBYSTATICINSTANCESAMPLE
パターン3のサンプル

¥COLLECTIONPERSISTANCESAMPLE
パターン4のサンプル

¥NESTEDPROPERTYESSAMPLE
ネストされたプロパティの永続化サンプル

¥NONPERSISTEDPROPERTYESSAMPLE
プロパティ値を保存しないサンプル

はじめに

前々回の記事で説明したように、デザイン時のコントロールの設定を変更するには、設定値を永続化できないと意味がありません。そこで、設定値の永続化について説明する前準備として、今回はオブジェクトのシリアライズに関して説明しました。

今回は、ようやく自作コントロールを Visual Studio .NET (以下 VS.NET) 上でカスタマイズした場合の設定値を永続化する方法について説明してゆきます。

設定値を 永続化する方法

VS.NETでのコントロールの設定を永続化するには、以下の方法があります。

- ①変更を VS.NET が自動生成するコードに反映する (自動生成するコードを変更する)
- ②設定をファイルに保存する
- ③設定をレジストリに保存する

③設定をレジストリに保存する

VS.NETに付属する標準コントロールは、①の方法で設定が保存されます。また、②と③の方法はアプリケーションでは一般的ですが、コントロールはアプリケーションで使用されるものであるため、あまり一般的な方法とはいえないかもしれません。そのため、ここでは①の方法 (コードの自動生成による設定の保存) に絞って解説することになります。

デザイン画面とInitialize Componentメソッド

VS.NETでフォームを新規作成すると、新しいクラスのコンストラクタとともに、以下のメソッドが必ず生成されています。

```
private void InitializeComponent()
```

この部分はC#では#region～#endregionブロックの間に挟まれているため、デフォルトの設定のままだと、隠れて

います^[注1]。

この部分をVS.NETのソースエディタ上の田のような形のアイコンをクリックして表示してみると、設定内容に伴ったソースコードが自動生成されていることがわかります(図1)。

自作のクラスをエディタで一から作成した場合でも、このInitializeComponentメソッドを用意しておけば、VS.NETで作成したクラス同様に変更結果がデザイン画面に反映されます。

Formクラスの派生クラスをVS.NETで作成した場合などは、このメソッドはコンストラクタで呼び出されています。しかし、実際にはこのメソッドはどこからも呼び出されなくてもデザイン画面に反映されます。反対に、コンストラクタにいくらコードを追加しても、InitializeComponentメソッドに変更がない場合は、VS.NETのデザイン画面には反映されません。

自動生成される内容

まず自動生成されるコードを確認します。Windowsアプリケーションを作成して、自作のコントロールを追加し、必要であれば、プロパティウィンドウでいくつか操作をした後、InitializeComponentメソッドの中身を確認してみます。以下の説明では、必要に応じて確認しながら読み進めてください。

VS.NETのデザイン画面で変更を加えた際に、InitializeComponentメソッドの中に自動生成される内容はさまざまですが、大きく分けると以下のようなパターンになります。それぞれのパターンごとに、パターンに属する型の例、共通する特徴、生成されるコードのサンプルを記します。

図1：VS.NETによって自動生成されたソースコード

```

41:         /// <summary>
42:         /// デザイン サポートに必要なメソッドです。このメソッドの内容を
43:         /// コード エディタで変更しないでください。
44:         /// </summary>
45:         private void InitializeComponent() {
46:             System.Resources.ResourceManager resources = new System.Resources.
47:             ResourceManager(typeof(Form1));
48:             this.cpsControl1 = new CollectionPersistenceSample.CPSControl();
49:             this.SuspendLayout();
50:             //
51:             // cpsControl1
52:             //
53:             this.cpsControl1.IntCollectionWithAddMethod.Add(0);
54:             this.cpsControl1.IntCollectionWithAddMethod.Add(0);
55:             this.cpsControl1.IntCollectionWithAddMethod.Add(0);
56:             this.cpsControl1.IntCollectionWithAddRange.AddRange(new int[] {
57:                 0,
58:                 0,
59:                 0});
60:             this.cpsControl1.Location = new System.Drawing.Point(0, 0);
61:             this.cpsControl1.Name = "cpsControl1";
62:             this.cpsControl1.Size = new System.Drawing.Size(75, 23);
63:             this.cpsControl1.TabIndex = 0;
64:             this.cpsControl1.Text = "cpsControl1";
65:             this.cpsControl1.Wrappper = ((CollectionPersistenceSample.CollectionWrapper)
66:             (resources.GetObject("cpsControl1.Wrappper")));
67:             //
68:             // Form1
69:             //
70:             this.AutoScaleBaseSize = new System.Drawing.Size(5, 12);
71:             this.ClientSize = new System.Drawing.Size(292, 273);
72:             this.Controls.Add(this.cpsControl1);
73:             this.Name = "Form1";
74:             this.Text = "Form1";
75:             this.ResumeLayout(false);
76:         }
77:         #endregion
78:         /// </summary>

```

パターン 1 定数そのまま保存される

例：string、int、float、enumなど

特徴：単純なValue型

サンプルコード

```
this.label1.Text = "label1";
```

パターン 2 新規インスタンスがコンストラクタで指定される

例：Font、Size、Pointなど

特徴：コンストラクタのある型

サンプルコード

```
this.comboBox1.Size = new System.Drawing.Size(121, 20);
```

パターン 3 staticなメソッド、メンバ、プロパティなどで インスタンスが指定される

例：Colorなど

特徴：コンストラクタのないStructureやClassなどの型

注1) この部分の設定の変更方法は2003年12月号の本連載を参照してください。