コンポーネント化のメリットと手法を追求する

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

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

¥NONPERSISTEDPROPERTIESSAMPLE
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、前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

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します。

デザイン画面とInitializeComponent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

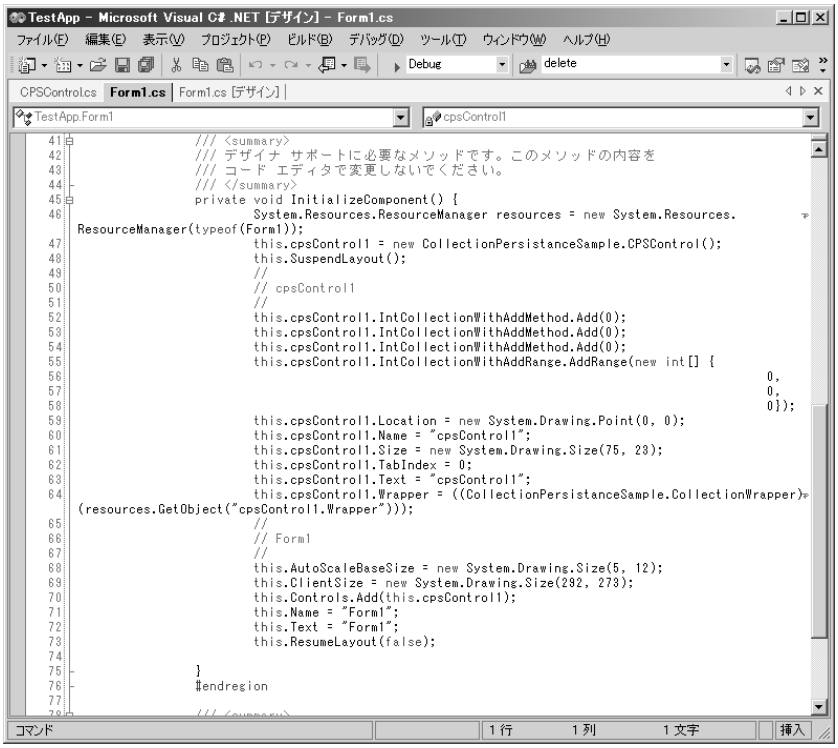
います[注1]。

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。

図1：VS.NETによって自動生成されたソースコード



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。

パターン1 定数が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。