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3回 デザイン時に自作コントロールの設定を変更する方法

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**

・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¥NETARCH01ディレクトリに収録しています。

¥MYPOINT1
「プロパティへの文字列の代入」サンプル
¥MYPOINT2
「ネストしたプロパティの展開」サンプル
¥MYPOINT3
「独自ドロップダウンUIでのプロパティ操作」サンプル
¥MYPOINT4
「ダイアログでのプロパティ操作」サンプル
¥MYPOINT5
「カスタムエディタ（デザイナ）の表示」サンプル
¥REFLECTION
「デザイン時のコードを別アセンブリにする」サンプル

## はじめに

今回は、自作のコントロールをVisual Studio .NET（以下VS.NET）で使用する場合に役立つ方法を説明します。

コントロールを利用するときは、必ず設定を変更し、自分の使用目的に合うようにします。その際、コードでの設定変更以外に、VS.NETでのデザイン時にも各種の設定値を変更できるようになっていれば格段に便利なコントロールとなります。

また、コード上での値の設定は、VS.NET上のデザイン画面に基本的には反映されていないため、プログラムを実際に実行してみなければ設定後の動作がわからないというデメリットもあります。

デザイン時にプロパティを設定する方法として、主に以下の3つがあります。

①VS.NETのプロパティウィンドウを使用する
②VS.NETから、独自のエディタ（デザイナ）を起動して値を設定する
③VS.NETのデザイン画面でマウスを使用して直接値を変更できるようにする

③の方法は、実際には②のデザイナを実装／使用することでより細かい設定が可能です。詳細は後述しますが、②のデザイナを実装する場合は、コントロールを配置するコンテナとして通常のフォームを使用するため、通常のフォームのプログラミングで済みます。これに対して、③の方法をとる場合、VS.NETをコンテナとするため、若干複雑なプログラミングを行なわなければなりません。そのため、あえて③は避け、ここでは①と②の実装方法に関して説明してゆくことにします。

## プロパティウィンドウでの操作

まずは、プロパティウィンドウでの操作方法に関して見てみましょう。

例として、標準Labelコントロールのプロパティを取り上げます。フォームのデザイン画面で目的のコントロール

を選択して、右クリックすると表示されるコンテキストメニューから［プロパティ］を選択すると、プロパティウィンドウにそのコントロールのプロパティが表示されるようになっています（図1）。

ここにはプロパティに関するさまざまな情報が表示されるほかに、それぞれのプロパティを簡単に設定するための工夫がされています。プロパティの設定には大きく分けて、以下のような特徴があります。

**特徴1▶文字列で値を代入できるもの**

該当するプロパティ例：Text、Size、Fontなど

**特徴2▶プロパティの横に、十字のアイコンが表示され、ネストされたオブジェクトを展開できるもの**

該当するプロパティ例：Location、Size、Fontなど（図2）

**特徴3▶独自のUIで値を変更できるもの**

該当するプロパティ例：BackColor、Dockなど（図3）

**特徴4▶ダイアログが表示され、値をダイアログ上で変更できるもの**

該当するプロパティ例：Fontなど（図4）

以降で、上記のそれぞれの特徴に関して、5つのサンプルプログラムを利用して実装方法も含めて説明してゆきます。

## プロパティへの文字列の代入

実は標準コントロールにあるほとんどのプロパティ、特にプロパティウィンドウに表示されるプロパティは、文字列で値を設定することが可能です。では、Fontプロパティ やSizeプロパティなど、String型ではないプロパティの値を文字列で表わすにはどうすればいいのでしょうか？

.NET Frameworkでは型変換を使用します。型変換にはその名もずばり、TypeConverterクラスを使用します。

図1：プロパティウィンドウの表示

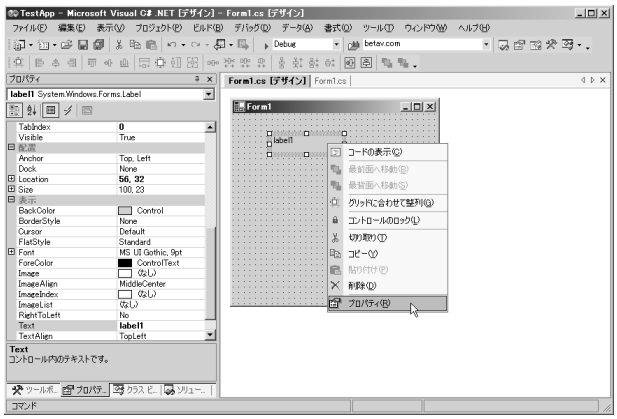

図2：ネストされたプロパティの展開（Fontプロパティ）

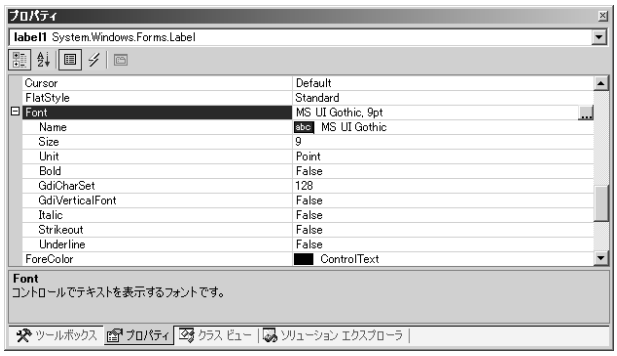


図3：独自UIでのプロパティ操作（Dockプロパティ）

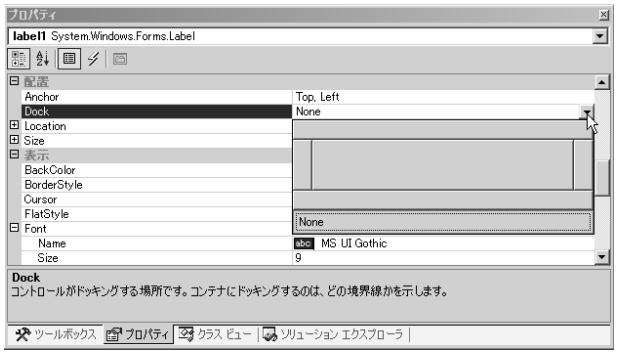


図4：ダイアログでのプロパティ操作（Fontダイアログ）