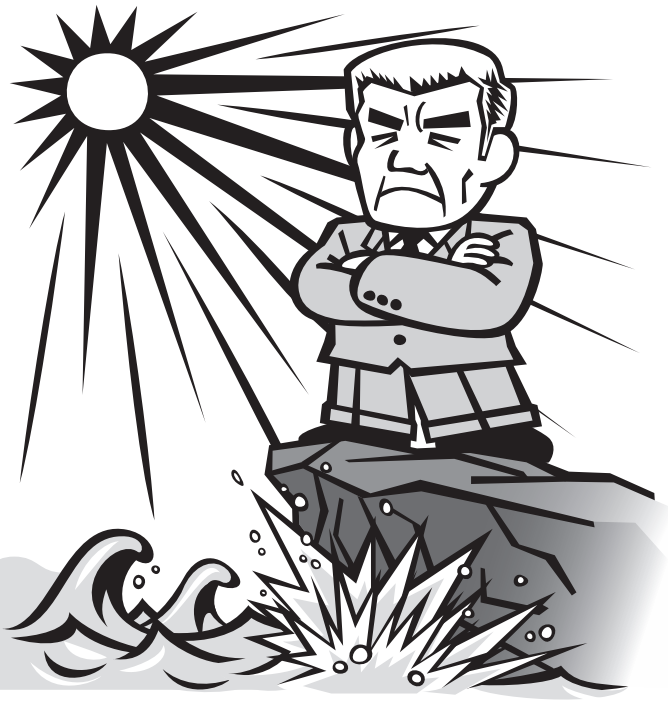
頑固一徹の 天気晴朗なれど波高し

西田 雅昭
NISHIDA, Masaaki



Visual Basic .NET奮戦記

第4回 VB6.0ユーザーがVB.NETで注意したいこと

Illustration : Toshiyuki Ido

level

1 2 3 4 5

Technology Tools

- ☑ Visual Basic
- ☐ Visual C#
- ☐ Visual C++
- ☐ SQL Server
- ☐ Oracle
- ☐ Access
- ☐ ASP.NET
- ☑ Other:
  - Visual Basic 6.0
  - Visual Studio 2003

Samples

我輩の名前は「頑固一徹」。もう10年以上Visual Basicを使ってカスタムプログラムを書いてきた我輩だが、VB.NETを使うハメに。
まずは練習のため、作り慣れた顧客管理システムをVB.NETで作ってみようと思い、前回は「メニュー」フォームを作成した。今回は、システムの構築をお休みし、これまで試してきたなかでわかったVB.NETでの変更点を、我輩流に紹介することにする。

エラーを恐れずコーディングすべし

今までVB6.0と同じ感覚でVB.NETをあれこれ触わってみたが、ちょっとした操作の違いを除けば、別に考えを変えなくても、VB6.0と同じように使えるようである。「VB.NETらしいプログラミングをしろ」などと言っていた西田雅昭なんかくそ食らえだ。

最初はとまどったが、ヘルプの使い方を勉強したので、それほど苦労することはなくなってきた。

ただひとつ、フォームから別のフォームを開くには、

```
frmMyForm = New frmMenu
```

のように、開くフォームのインスタンスを作らなければならないのにはまいった（前号を参照）。まあ考えてみれば、自分で作ったクラスを使うときと同じなので、ちょっと面倒なだけなのだが。

あとは、VB6.0と同じ感覚でコードを書いていけばよい。たとえば、

```
me.caption = "メニュー"
```

と記述して、[Enter]キーを打った瞬間、青色の波線が出てエラーを教えてくれる。この波線のところをマウスでポイントすると、エラーの内容がすぐ現われる（図1）。実行するまでエラー

図1：波線をポイントするとエラーの内容がわかる

```
Me.caption = "メニュー"
    'caption' は 'Test1.Form1' のメンバではありません。

End Sub
```

がわからないVB6.0と比べて、はるかに便利である。

ここでは、「Caption」がFormのメンバではない、というのだから、「Form」のプロパティウィンドウを調べてみればよい。「Caption」プロパティの代わりに、「Text」プロパティを使えばよいことがすぐわかる。

オブジェクトのメンバについては、「me.」と記述した瞬間にメンバの一覧が出るので、「Caption」プロパティがないことがわかったら、別のプロパティを調べてみればよいだろう。

オブジェクトのメンバ以外に関しても、「パラメータヒント」（関数やメソッドのパラメータの一覧）が常に表示されるので、これを利用すればよい。

また［Ctrl］+［Space］キーで、入力候補の一覧を見ることもできるので、たいていのことは、一覧からの選択で済んでしまう。

これらの機能を利用すれば、VB.NETの変更点など恐るるに足らず。かまわずに、VB6.0と同じ感覚でどんどんコーディングを進めればよい。「VB.NETらしいコーディング」などというものは、暇なときに、少しずつ勉強すればよいのだ。

たとえば、「MessageBox」クラスを使わなくても、「MsgBox」関数で用は足りる。「MsgBox」関数でも、VB.NETの便利な「パラメータヒント」を利用することができる。要は、VB.NETの便利な機能は最大限に利用し、VB6.0の感覚でコードを書くことである。このように考えると、VB.NETは、いろいろ便利な機能があって、結構使いやすいものである。

名前が変わったコントロールに気をつけろ

と言っても、いちいちヘルプのご厄介になるのは面倒だ。この数か月で書き溜めたコントロールに関する我輩のメモを、少し紹介することにしよう。最低、これぐらいを記憶しておけば、後は、ヘルプを参照しながら、楽にコーディングできるだろう。

まずは、名前の変わったコントロールをアルファベット順に列挙しておく（表1）。それにしてもMSの勝手気ままな名前の変更には困ったものだ。

なくなったコントロールもあるぞ

次にVB.NETでなくなったコントロールと、それに代わる利用法をまとめておく。

表1：VB.NETで名前の変わったコントロール

VB6.0	VB.NET	役割
CommandButton	Button	プロセスを開始、停止、または中断するために使用する
DTPicker	DateTimePicker	ユーザーが日付または時刻を選択できるグラフィカルなカレンダーを表示する
Frame	GroupBox	ラベルの付いたスクロールできないフレームに、コントロール（オプションボタンなど）のセットをグループ化する
Menu	MainMenu	メニュー
MonthView	MonthCalendar	ユーザーが日付の範囲を選択できるグラフィカルなカレンダーを表示する
OptionButton	RadioButton	オンまたはオフにできるボタンを表示する
Slider	TrackBar	ユーザーが目盛り上でつまみを動かして値を設定できるようにする
TabStop	TabControl	グループ化されたオブジェクトを効率的に整理してアクセスするためのタブ付きページを提供する

イメージ表示

「Image」コントロールがなくなった。代わりに「PictureBox」コントロールを使う。

「Image」コントロールのメンバは、すべて同じ名前で「PictureBox」コントロールでサポートされている。ただし、「Image.Stretch = True」は、

```
PictureBox.SizeMode = StretchImage
```

に、「Image.Stretch = False」は、

```
PictureBox.SizeMode = Normal
```

に書き直さなければならない。

グラフィック関連

「Shape」コントロールと「Line」コントロールがなくなった。代わりに「GDI＋」を使う。

これには本当に頭にきた。VB6.0とまったく違うからだ。基本的な使い方はヘルプを見ればわかるが、GDI＋は随分と多機能なので、使いこなすには時間がかかりそうだ。ここでは、試しに我輩が書いてみたコードを紹介しておく（図2・リスト1）。

VB6.0とまったく違うことがわかるだろう。グラフィック関連のコードを書くことが多い場合は、GDI＋について少し勉強しておいたほうがよいかもしれない。ここではコードの説明はしないが、わからない方は、ヘルプの「GDI＋によるグラフィカルイメージの作成」を参照していただきたい。

なお、水平線と垂直線を描きたい場合は、GDI＋を使わずとも「Label」コントロールを利用すればよい。「Label」コントロールで、「Width」プロパティを「1」か「2」にすれば垂直線、「Height」プロパティを「1」か「2」に