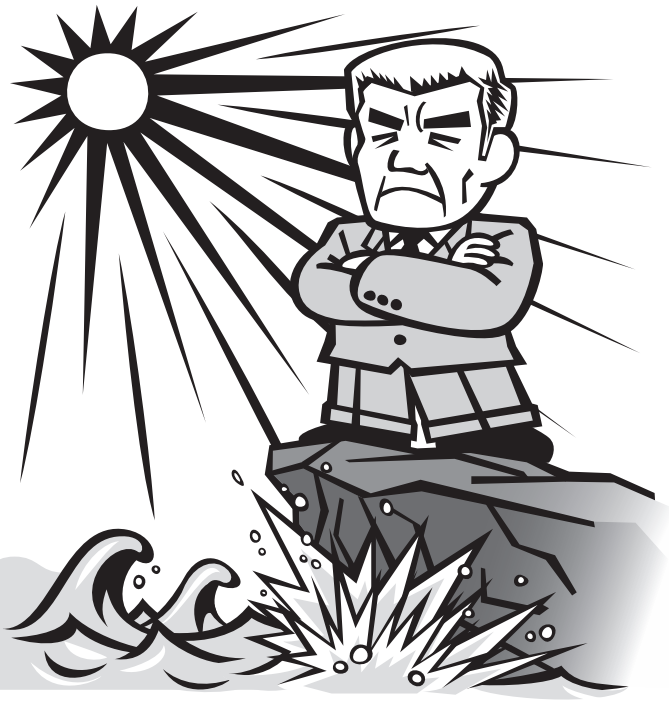
頑固一徹の 天気晴朗なれど波高し

西田 雅昭
NISHIDA, Masaaki



Visual Basic .NET奮戦記

第5回 コンボボックスとリストボックスの変更点

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。
前回までで、注意すべきVB.NETでの変更点がわかり、やっと顧客管理システムを作れると思ったのも束の間、コンボボックスとリストボックスの使い方がひどく変わっていることに気づいた。癪に障るので、VB6.0で書いて、アップグレードしてみたが、それでも動かないところが残った。頑固な我輩だが、動かないものは仕方がない。この2つのコントロールの、旧バージョンとの違いを調べることに相成った。今回はその成果を紹介する。

コンボボックスの何が変わったのか？

まずは、コンボボックスの変更点から紹介する。

項目の追加

VB6.0では、コンボボックスの項目は文字列であり、「AddItem」メソッドを使って項目を追加したり、「RemoveItem」メソッドを使って項目を削除できた。また、「Clear」メソッドで、すべての項目を消すこともできた。

VB6.0

```
ComboBox1.AddItem "西田　雅昭"
```

“何でもオブジェクト”のVB.NETでは、面倒なことにコンボボックスの項目は「ComboBox.ObjectCollection」というオブジェクトコレクションになっている（その代わり、文字列以外に、イメージやプログラマが用意したビジネスオブジェクトも使えるが）。項目の処理には、この「ComboBox.ObjectCollection」クラスのメソッドを使うことになる。

したがって項目を追加するには、「ComboBox.ObjectCollection」の「Add」メソッドを使わなければならない。それには、「ComboBox」オブジェクトの「Items」プロパティで「ComboBox.ObjectCollection」を取得してから、「Add」メソッドを使う必要がある。

```
ComboBox1.Items.Add("西田　雅昭")
```

この「Add」メソッドには戻り値があり、現在追加した項目の位置（Index）を知ることができる。

また、VB6.0では、次のように、「AddItem」メソッドに引数を追加して、指定の位置に項目を挿入できた。

VB6.0

```
ComboBox1.AddItem "西田　雅昭", 3
```

しかしVB.NETの「Add」メソッドには、このような機能がないので、「Insert」メソッドを使って、

```
ComboBox1.Items.Insert(3, "西田　雅昭")
```

と記述しなければならない。引数の順序が変わったのも癪に障る。

なお、「Add」メソッドは、実行するたびにコンボボックスを再描画する。そのため、多くの項目を追加する場合には、システムのパフォーマンスに影響がある。そこで「BeginUpdate」メソッドを使用して、再描画を中断し、すべての項目の追加を終了してから、「EndUpdate」メソッドで再描画を再開するとよい。

```
With ComboBox1
  .BeginUpdate()

    多数の項目の追加処理

  .EndUpdate()
End With
```

VB.NETで便利になったこともある。配列の値を項目として追加する場合である。VB6.0では、

VB6.0

```
For intKazu = 0 To 20
  ComboBox1.AddItem astrName(intKazu)
Next
```

のように「For」文で記述したが、VB.NETでは、

```
ComboBox1.Items.AddRange(astrName)
```

と1行で済んでしまう。非常に簡単である。「AddRange」メソッドは、すべての項目の追加が終わってから再描画をするので、再描画に関する配慮が必要ないのがうれしい。

項目の削除

項目の追加の場合と同じように、VB6.0の「RemoveItem」メソッドの代わりに「ComboBox.ObjectCollection.RemoveAt」メソッドを使わなければならない。

```
ComboBox1.Items.RemoveAt 1
```

VB.NETでは、位置（index）を指定する代わりに、項目自体を指定して削除する「Remove」メソッドも存在する。

```
ComboBox1.Remove("西田　雅昭")
```

「Remove」メソッドを使うと、ユーザーが選択した項目を削除する場合にも、次のように、簡単にわかりやすい形で記述できる。

```
With ComboBox1
  .Items.Remove(.SelectedItem)
End With
```

なお、すべての項目を削除する場合には、VB6.0と同じ「Clear」メソッドを使用して、次のように記述すればよい。

```
ComboBox1.Items.Clear()
```

また、VB.NETでは、デザインモードで項目を処理する際に、便利な機能がある。コンボボックスを選択しておいて、プロパティウィンドウで「Items」プロパティを選択すると、省略記号ボタン（図1）が現われる。これをクリックすると「文字列コレクションエディタ」ダイアログボックス（図2）が開く。ここではコンボボックスの内容の一覧を見ることができるし、複数の項目を入力したり訂正したりすることもできる。VB6.0の「List」プロパティと似た機能だが、こちらのほうが使いやすい。

なお、これまでに紹介した「Add」「Insert」「Remove」「RemoveAt」などのメソッドや「Items」プロパティは、VB.NETでは、いろいろなコレクションの操作に使われるようなので、使い方を覚えておくとよい。

Memo

VB.NETでは、プロパティに「(コレクション)」と表示されている場合には、ほとんどの場合、なんらかのダイアログボックスが開く。「(コレクション)」という文字にぶつかったら、必ず、右にある省略記号ボタンをクリックしてみるとよい。多くの場合、便利な機能を知ることになる。

コンボボックスの外観

VB6.0では、「Style」プロパティで、

図1：省略記号ボタン

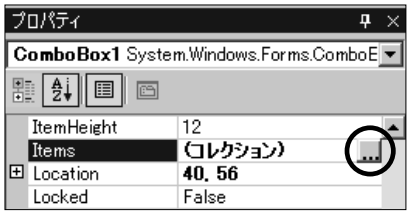


図2：文字列コレクションエディタ