Visual Basic .NETのツボ 2003

第15回 コレクションの作成と利用－その2－

西田 雅昭
NISHIDA, Masaaki

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\TUBOディレクトリに収録しています。

\COLLECTION
前回作成したコレクションサンプル（CassTest1プロジェクト）

\SORTEDLIST
今回作成した「SortedList」コレクションを使ったサンプル

最初からお詫びで申し訳ありません。前回、「ユーザーインターフェイスとビジネスルールの分離」「ネーミング」についてお話しすると言いましたが、この説明は次回に繰り越し、今回は引き続きコレクションについてお話しすることにします。「コレクションというやつを使っても、たいしたメリットがないではないか」というご意見をいただいたので、今回は「SortedList」コレクションを利用して、そのメリットについてみてみましょう。

はじめに

前回、ヘルプの「System.Collections名前空間」（図1）をお見せした際に、「CollectionBase」コレクション以外のコレクションクラスは無視していました。Visual Basic 6.0（以下VB6）の「Collection」クラスを知っている方には、「CollectionBase」コレクションが一番理解しやすいと思ったからです。

しかし、「System.Collections名前空間」には、「CollectionBase」コレクション以外にも便利な機能をあらかじめ備えたクラスが存在します。VB6以前は、配列を利用して複雑な構造を作成するのはなかなか難しいことでした。しかし、.NET Frameworkの世界では、複雑な構造でも簡単に作成することができます。

ヘルプを見ていただければわかることですが、コレクションの機能一覧と、ごく簡単な説明を表1に示しておきます。プログラミングに慣れた方なら、この説明を見ただけで、これらのコレクションを使いたくなるでしょう。

一般的なデータ処理を行なう場合、表1のなかでもっとも有効なのは「SortedList」コレクションでしょう[1]。今回は、この「SortedList」コレクションの使い方を説明することにします。

▼1　機能が多いということは、反面、パフォーマンスがよくないということになるが、最近のマシンは高速になっているので、よほど特別な場合を除いて問題はないと思う。特に高速性が要求される場合には、「ArrayList」や「Hashtable」を使えばよい。

図1：System.Collections名前空間のヘルプ

表1：便利なコレクション

コレクション（クラス）名	説明
ArrayList	必要に応じてサイズが動的に増加する配列のようなコレクション。任意の位置へデータを追加したり、削除したりできる
Hashtable	キーのハッシュコードに基づくコレクション。高速検索が可能
CaseInsensitiveHash CodeProvider	文字列の大文字と小文字を区別しないハッシュコードを提供する
DictionaryBase	厳密に型指定された、キーと値の組み合わせのコレクション
SortedList	キーによって並べ替えられ、キーとインデックスを使ってアクセスできるコレクション。「ArrayList」コレクションと「Hashtable」コレクションの両方の機能をもつ
BitArray	ビット値の小型の配列
Queue	オブジェクトの先入れ先出しをする（FIFO）
Stack	オブジェクトの単純な後入れ先出しをする（LIFO）

表2：「SortedList」コレクションの「Add」メソッド

機能	指定したキーおよび値をもつ要素を「SortedList」コレクションに追加する	
構文	Overridable Public Sub Add(ByVal key As Object, _ ByVal value As Object) Implements IDictionary.Add	
パラメータ	key	追加する要素のキー
	value	追加する要素の値

「SortedList」コレクションの利用

西田式は、常にコードを記述して実験することを主眼としています。前回、「CollectionBase」コレクションを使って実現した機能を、「SortedList」コレクションを使って実現してみましょう。

本連載では、ひとつのプロジェクトにコードを追加する形で話を進めています。今回は、前回までのプロジェクトに、クラスをひとつ追加して「SortedList」コレクションを利用する処理コードを記述してゆきます。前回までのプログラムは、付録CD-ROMに収録しているプロジェクトを参照してください。

前回作成した「ClassTest1」プロジェクトをVisual Studio .NET（VS.NET）で開き、メニューから、

［プロジェクト］-［クラスの追加］

と選択します。すると、「新しい項目の追加」ダイアログボックスが現われるので、「ファイル名」テキストボックスに「colSortList」と入力し、［開く］ボタンをクリックします。

コードウィンドウが開くので、まず、

```
Dim mcolSortList As New SortedList()
```

と記述してください。

前回のコレクションは「継承」（Inherits CollectionBase）で始まっていましたが、今回は、「Class…End Class」のプログラムブロックの中で、「SortedList」コレクションのインスタンスを作ることから始めました。

VB.NETでは、このプログラムブロックの中に、複数のクラスを記述することもできますし、「Form」クラスを記述することも可能です。

◉メソッド「fPersonAdd」の作成

最初に作成するのは、「SortedList」コレクションにクラスを要素として追加するメソッドです。クラスを要素として追加するには「SortedList」コレクションの「Add」メソッドを使います（表2）。

前回のプログラムでは、要素の読み出しはインデックスを利用していました。「SortedList」コレクションは、インデックスとキーによる読み出しの両方をサポートしているので、要素とキーをセットで追加する自作メソッド「fPersonAdd」を作成することにします。

自作メソッド「fPersonAdd」のコードについてはリスト1を参照してください。

検索などを行なうキーには「ふりがな」を使うことにしました。「SortedList」コレクションのキーは重複を許さないので、重複のチェックが必要です。

「SortedList」コレクションの「Contains」メソッドは、「SortedList」コレクションに特定のキーが格納されているかどうかを判断し、すでに同じキーが存在しているときは“ True ”を返し