新連載

bject BLV オブジェクトの散歩道

例題でわかる！ .NET Framework

第1回 「Stringを考える」

吉田 弘一郎 *YOSHIDA, Koichiro*

本連載ではさまざまなコードを示しつつ、初歩の初歩からじっくりと.NET Frameworkにインプリメントされているクラスの機能を解説してゆきます。

まずは、今回のお題である「String」と.NETの関係性について考えてみることにしましょう。

String操作の標準化

どのようなプログラムを作るにせよ、何らかの文字列（String）の操作が必要になります。昔ながらの書式付きの印刷出力から、モダンなGUIのコンポーネントに表示する場合まで、非常に幅広い分野で文字列を操作することになるのです。ところが、この文字列の操作ほどプログラム言語やライブラリに依存するものはありません。とにかく、それは多彩であり、統一がとれていません。マイクロソフトによる.NETの導入の影響は、この“文字列”の扱いにも顕著に表われているのです。

まず.NETの特徴は次の2点であることを強調したいと思います。

①プラットフォーム独立性
②プログラム言語独立性

もちろん、前者に関してはWindowsの域を出ていないどころか、現状では「.NET対応Windows OS」も存在しません。つまり、Windowsをインストールしただけでは.NETすることはできず、Windowsのアップデートという形で後から付け加えることになります。しかし、これは時間が解決することでしょうから、うるさいことを言うのは控えましょう。

それよりも、後者のプログラム言語独立性が重要なのです。つまり、少なくとも.NETの世界においては、プログラム言語に依存する文字列の扱いに差はなくなったのです。正確には「高度にコンパチになった」というべきかもしれません。.NETの一員となった個々の言語から見れば、「.NET標準仕様を採用するハメになったが自らも標準になった」ということになります。そして、その影響は文字列操作関係で非常に顕著であったのです。

今回はこの状況をVisual Basicの立場から見ることからはじめます。

それでは、簡単な例題を見てみましょう。今回紹介する例題は、すべてVisual Basic.NETのコンソールアプリケーション用のコードとなります。コンソールにしたのは、話を簡単にするためです。

本稿で前提となるもの

OS	Windows 2000 Professional（SP3）以降
開発環境	Visual Studio.NET .NET Framework 1.0.3705.288（SP2） Internet Explorer 6.0.2800.1106

初級　中級　上級

例題1：文字列をコンソール上に書き出す

Visual Basic.NETのコンソールアプリケーションプロジェクトで、「Hello, VB.net」してみましょう（リスト1）。.NETらしくConsole.WriteLineメソッドを用いています。

実行結果は図1のとおりです。Visual Studio.NETの中から走らせたので、「Press any key to continue」という行も表示されます。

参考までに、このプログラムをC#にするとリスト2のようになります。ここで注目すべきは、

リスト1：例題1（VB.NET）

```
Module Module1

    Sub Main()
        Console.WriteLine("Hello, VB.net.")
    End Sub

End Module
```

リスト2：リスト1をC#に書き換えた結果

```
using System;

class Class1
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello, C#");
    }
}
```

リスト3：リスト1をJ#に書き換えた結果

```
import System.Console;

public class Class1
{
    public static void main(String[] args)
    {
        Console.WriteLine("Hello, J#");
    }
}
```

```
Console.WriteLine("...")
```

の部分は、完全に両言語共通であるという点です。それもそのはず、同じクラスの同じ関数（メソッド）を用いているからです。Visual Basicには、Basicのprint文もあります。ファイルに書き込む場合に使いましたね。しかし、ここでは、.NETのConsoleクラスのWriteLineというメソッドを使っています。蛇足ながら、J#ではリスト3のようになります。

要するに、各々の言語の文法はその言語特有のものですが、クラスライブラリのほうは言語にはよらないのです。みなさんがVisual Basic.NETを用いるために習得された.NETのクラスの知識と経験は、C#やJ#でもそのまま役に立つのです。ですから一層、クラスライブラリの習得が重要になります。

それなら、ここでConsoleクラスを少々覗いてみようではありませんか。方法は簡単で、開発環境のエディタ上で「Console」と記述し、続けてピリオド（.）を記述した状態でちょっと待っていると、図2のように「使用可能なメンバのリスト」が出てきます。メンバには関数もあればプロパティもあります。ここでは、ポップアップしたリストをスクロールして、WriteLineを見えるようにしました。すると、上のほうにReadLineもあります。次の例題では、このReadLineも使うことにしましょう。

「使用可能なメンバのリスト」でReadLineをセレクトして［Enter］キーを押すと、エディタ上で、

```
Console.ReadLine()
```

となります。さらに、ReadLineの詳細を知るためには、ReadLine()の直後にカーソルをもってゆきます（図3）。これにより、ReadLineがString型の変数を返すメンバ関数であることがわかりました。なお、Visual Basic.NETの場合には、ReadLine()でもReadLineでも構いません。

図1：リスト1実行結果

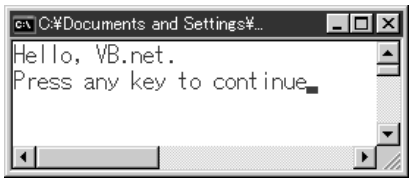

図2：使用可能なメンバのリスト

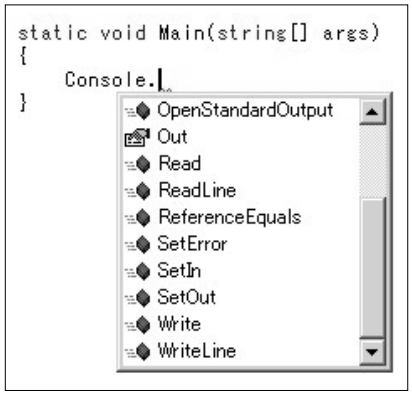

図3：ヒントを表示