特集1

Visual Studio 2005 β2

Visual C# 2005 に見る新機能

開発環境の充実と開発言語としての深化

こだか かおる
KODAKA, Kaoru

Level

1 2 3 4 5

Technology Tools

- ☐ Visual Basic
- ☑ Visual C#
- ☐ Visual C++
- ☐ SQL Server
- ☐ Oracle
- ☐ Access
- ☐ ASP.NET
- ☑ Other:
  Visual Studio 2005 ベータ2

Samples

はじめに

.NET Frameworkのバージョンにあわせ、C#もバージョン2.0になりました。新しい機能については、本誌連載「アドバンスド Windowsプログラミング」の前月号で紹介しています。もちろん、誌面の都合もあったため、すべてを紹介することはできませんでした。新しい言語仕様から紹介できたのは、「Anonymous Method」と「Nullable」のふたつだけです。

今回、ナイスタイミングで、再度C# 2.0の新機能について紹介させてもらう機会をいただきました。

連載のほうでは紹介しきれなかったため、ちょっと残念な思いをしていたわけですが、その無念はこの場で晴らすことにしたいと思います。さらっと取り上げるのではなく、サンプルコードを中心に少々深く掘り下げながら見てゆくことにしましょう。

C# 2.0の言語仕様の変化

言語仕様の変化といっても、新しい機能が追加されることがほとんどです。中には、今までのプログラミングスタイルを大幅に変えてしまうものから、ちょっとしたプログラムコードの省略ができるようになるものまで、いろいろな種類があります。

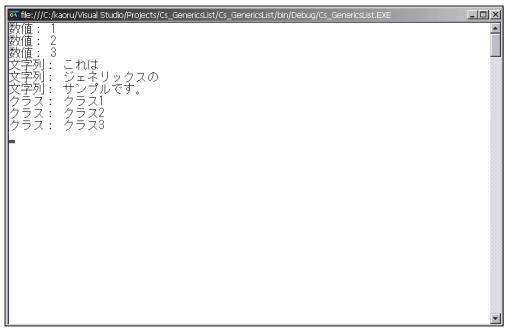
C# 2.0における前者の代表格は、やはり「ジェネリックス」でしょう。これを外すわけにはいきません。本当は連載の中で紹介したかったのですが、記事にまとめるには少々調査が足りませんでした。間違いをお伝えするわけにはいかないので、改めて調査を行ない、満を持しての紹介となります。今回の記事では、ジェネリックスのほかにも、ジェネリックスとなじみ深い関係にある「イテレータ」、さらに「静的クラス」を取り上げることにします。

ジェネリックスの使い方

さて、さっそくジェネリックスについて見てゆくことにしましょう。ジェネリックスは“総称”などと訳されます。あまりなじみのない言葉ですが、辞書で引いてみると「いくつかの物をひとつにまとめて呼ぶこと。また、その呼び名」などとあります（三省堂「大辞林 第二版」より）。辞書で言うところの「いくつかの物」というのは、ジェネリックスではintやstringといった「型」のことです。すなわち、型をまとめて使うための仕組み、と理解しておけばよいでしょう。

例として、ジェネリックスを利用するサンプルを作ってみました（図1）。

図1：ジェネリックスサンプル



サンプルコードを追いかけながら解説します（リスト1）。ここで利用しているのは、ジェネリックス型のひとつ「List<>」です。Listは、.NET Framework 1.xでのArrayListと同様のコレクションで、配列と同じように値を格納する入れ物です。名前の後に「<>」がついているのは、それがジェネリックス対応のListであることを示しています。サンプルコードの最初の部分、

```
List<int> li = new List<int>();
```

では、List<>に格納する値の型がintであることを宣言しています。同時に、List<int>のインスタンスを生成し、変数「li」に代入しています。この宣言によって、liにはint型の値のみが格納できるようになるというわけです。

その後の処理は、値の追加、foreachによる列挙です。サンプルプログラムの続きで行なっているように、string型でも同じように処理することができます。

最後に、

```
List<GenericClass> ls =
 new List<GenericClass>();
```

と宣言しています。intやstringといったC#にもともと用意されている型（プリミティブ型とも言います）と同じように、ジェネリックスでは、ユーザーが新たに作成したクラスをも受け入れることが可能です。すべての型を同じように扱える、それこそが、ジェネリックスのジェネリックスたるゆえんと言えるでしょう。

見てきたように、ジェネリックスによって、どのような型でも受け入れる仕組みが実現できることがわかりました。ここで、ちょっと視点を変えて、ジェネリックスの存在理由について考えてみます。

「どのような型でも扱える」ということから、なんとなく、Object型を使った方法と同じように思われるかもしれませ

リスト1：ジェネリックスを使う

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Cs_GenericsList {
 class Program {
  static void Main(string[] args) {

   // int 型の List を生成
   List<int> li = new List<int>();

   li.Add(1);
   li.Add(2);
   li.Add(3);
   foreach (int var in li) {
    Console.WriteLine("数値 : {0}", var);
   }

   // string 型の List を生成
   List<string> ls = new List<string>();

   ls.Add("これは");
   ls.Add("ジェネリックスの");
   ls.Add("サンプルです。");
   foreach (string var in ls) {
    Console.WriteLine("文字列： {0}", var);
   }

   // GenericClass 型の List を生成
   List<GenericClass> lgc = new
List<GenericClass>();

   lgc.Add(new GenericClass("クラス1"));
   lgc.Add(new GenericClass("クラス2"));
   lgc.Add(new GenericClass("クラス3"));
   foreach (GenericClass var in lgc) {
    Console.WriteLine("クラス： {0}", var);
   }
  }
 }

 public class GenericClass {
  private string _name;

  public GenericClass(string arg) {
   _name = arg;
  }
  public override string ToString() {
   return _name;
  }
 }
}
```