επιστημηの

オブジェクト指向的夜話

C#から見た.NET Framework

第8回

リフレクションで遊んじゃえ

επιστημη
えぴすてーめー

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¥YAWAディレクトリに収録しています。

¥INSPECT
型情報の抽出

¥INVOKE
型情報を使ったインスタンスの生成／メソッド呼び出し

¥TARGET
お試しアセンブリ

¥TESTGEN
NUnit対応テストコード雛型生成機

ないものねだりのリフレクション

ご存知の方も少なくないでしょうが、僕はC++屋です。それも純粋C++：ANSI/ISOの定める標準C++がホームグラウンド。次第に形を現わしつつあるC++/CLIはいまだ様子見の状態。現時点では.NETはC#が一番じゃねぇの？　と思ってます。C#にはまだまだ途惑うことも多いけど、文法そのものは難解ではないので（C#を難しいといったらC++はどうなるよ!?）アタマの中にあるアイデアをコードに落とし込むのにはそれほど苦労しなくなりました。

C++屋の僕が違和感というかC++との相違というか、“しっくりしない”と感じているところがあります。言語とライブラリ、ここではC#と.NET FrameworkとがC++とは比べようもないほどに密着しているんです。たとえば文字列。C++では標準C++ライブラリが文字列クラス「std::string」を提供していますが、C++の基本的な構文／型の中には「std::string」は含まれていません。あくまで言語とは独立した存在です。ところがC#では.NET Frameworkが提供するSystem.Stringが言語と密着しています。アプリケーションの入り口からして

```
static public void Main(String[] args)
```

ですもの。Stringなくてはアプリケーションの入り口すら作れません。

ま、Javaでも似たようなもの。それはそんなものと納得しています。正式リリースが待ち遠しいVisual Studio 2005ではC#2.0になるのかな？ 新しいC#ではいよいよGenericsがサポートされ、アイデアとコードのギャップをより小さくしてくれるものと期待しています。

なんだかんだいってもやっぱり僕はC++が好き。「今どきガベージコ

レクタもないような言語かよ？」と言われようが、「まだヘッダなんか読んでるの？」とけなされようが、20年近く寄り添ってきた言語ですもの、愛着がありますからね。こうなりゃ最後まで添い遂げてやりますとも（どっちが先に倒れるか知らんけどね）。

それほどに惚れ込んだC++なんですが、やはり（不本意にも）.NETが羨ましく感じることもあります。そのひとつがリフレクション（reflection）。コンパイルされたアセンブリの中から、そこに納められたクラス名はもちろんのこと、各クラスが持つフィールド（メンバ変数）、メソッド（メンバ関数）、プロパティその他ありとあらゆる情報を取り出すことができ、さらにそれらの情報を基に動的にインスタンスを生成し、メソッドを呼び出すことができるらしい。このカラクリがあるからこそ、オブジェクトブラウザやインテリセンスが提供できるわけです。C++でこれを実現するにはソースコード（ヘッダ）を解析しなければならず、DLLだけでは不可能です（型情報が含まれていませんから）。C++には真似のできないリフレクションで遊んでみましょう（いつものようにマニュアル首っ引きで）。

アセンブリからモノを読み出す

お試しにひとつ、小さなアセンブリを作りました（リスト1）。定義されているのは「interface counter_interface」と、それを実装した「class counter」の2つ。これをコンパイルしてtarget.dllを作ります。

リスト1：サンプルとして用いるアセンブリのソース（target.cs）

```
namespace target {
  // インターフェイス
  public interface counter_interface {
    void increment();
    int data { get; set; }
  }

  // クラス
  public class counter : counter_interface {
    // フィールド（メンバ変数）
    private int value_;
    // コンストラクタ(引数なし)
    public counter() : this(0) { }
    // コンストラクタ（引数あり）
    public counter(int initial) { value_ = initial; }
    // メソッド（メンバ関数）
    public void increment() { ++value_; }
    // プロパティ（メンバ変数に見せかけたメンバ関数）
    public int data {
      set { value_ = value; }
      get { return value_; }
    }
  }
}
```

```
> csc /target:library target.cs
```

で、こいつの“はらわた”をリフレクションで掻き出してみようという試みです。

リフレクションに絡むクラス群は名前空間System.Reflectionに納められています。System.Reflection.Assemblyにアセンブリを読み込み、この中にある型の集合を取り出します。

```
using System;
using System.Reflection;

Assembly assembly =
   Assembly.LoadFrom("target.dll");
Type[] types = assembly.GetTypes();
```

これでtarget.dllに納められた型（Type）の配列がtypesに格納されます（Typeは名前空間Systemに属します）。あとはtypesの要素ひとつひとつについて、さらに詳しい型情報を引き出せばよさそうです。

```
foreach ( Type type in types ) {
  （略）
}
```

名前空間と名前

まずは小手調べ、型が属する名前空間と型の名前を取り出します。Typeにはプロパティ「Namespace」と「Name」があり、それぞれ名前空間と名前です。