bject BLVD オブジェクトの散歩道

例題でわかる！ .NET Framework

第11回 「Javaと.NETの実行速度」

吉田 弘一郎 *YOSHIDA, Koichiro*

Technology Tools

- ☑ Visual Basic .NET
- ☑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ☑ Other:
  - Visual C++ .NET
  - MingW
  - Java

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\SAMPOディレクトリに収録しています。

・SAMPO11.SLN
今回紹介したサンプルのソリューションファイル

\JAVA01
Java01.java
（Javaで記述した実行速度のテストコード）

VS .NET 2003とC# Builder

先日、何台もの英語版Windows 2000機のVisual Studio .NETをVisual Studio .NET 2003にバージョンアップしました。Visual Studio .NET 2003のベータ版から正式版へのバージョンアップは非常に厄介だったので、今回も気が進まなかったのですが、いつまでも延ばすわけにはゆきません。

結果は目下、2勝3敗。まったく何のトラブルもなかったのが2件です。これらは、いずれも3枚のCD-ROMを用いての話で、トラブルというのは「リブートせよ」の無限ループのことです。DVD-ROMに代えて、強引にWindows Component Updateを実行してなんとかなりました。こんな状況なので、日本語版Windows 2000機でのバージョンアップは、少々先になる見込み。

こうして、英語版Visual Studio .NET 2003を使い始めたのですが、面白いのはPreEmptive社の「dotfuscator」のコミュニティ版が付いてくることです。先月話題にした逆コンパイラを防止するツールです。これは誰でもダウンロードできるコミュニティ版なのですが、それをバンドルする「したたかさ」は賞賛に値します。逆コンパイラなんかにはまったく興味のなさそうなSUNにも見習ってもらいたいものです。

さて、Visual Studio .NET 2003をインストールすると、.NET Frameworkも1.1版になります。最近発売になったBorland社のC# Builderに含まれる.NET Frameworkも1.1版です。別のPCにWindows Server 2003を載せ、その上でC# Builderのエンタープライズ版の評価を始めたところです。実際、非常に興味深い製品ですので、評価記事をお楽しみに。

ところで、C# Builderには無料のパーソナル版もあって、これがまた無料とは思えぬ品質。今までのパーソナル版は、安っぽい感じがしないでもなかったのですが、このC# Builderのパーソナル版はまことに堂々としたものです。C# Builderパーソナル版のセットアップには.NET Framework 1.1も含まれているので、Visual Studio .NET 2003なしでも無料でVisualなC#開発を楽し

むことができます。これを見逃す手はないと思います。C# Builderのセットアップに関する注意事項としては、Javaがインストールされているのが前提になっているということです。私の場合、セットアッププログラムがJavaのインストールしてある場所を聞いてきてから、あわててJavaをダウンロードしてインストールしました。このC# Builderパーソナル版の登場で、C#する敷居の高さがなくなったように思えます。

JavaからC#への変換ツール

Java Language Conversion Assistant、略してJLCAのベータ版もMicrosoft社のサイトからダウンロードできます。これは、Visual Studio .NETの中で動くツールです。.NET Framework 1.0のVisual Studio .NETで動きました。早速、何年も前に作った結構大きなJavaアプリケーションのソースファイルのディレクトリを指定して変換してみました。結構時間をかけて、相当真面目な結果を出してくれたように思えます。興味深いのは、setX、getXという感じでペアがあると、気を利かせてこの「X」をプロパティに直してくれるのです。ベータですから、まだまだ完全ではありませんが、JavaからC#への変換は一挙に能率アップという感じです。正式版になれば、恐ろしいほどの威力を発揮するでしょう。いや、ベータ版でも、非常に役に立ちます。

Javaのソースがあればこの JLCAで簡単にC#に変換できるのです。ソースがなくても、同じ話です。C#からVisual Basic .NETへの変換ツールはありませんが、これも逆コンパイラを活用すれば可能ですから、大変な時代になったものです。プログラムの主役であった言語の絶対性が完全に崩れるわけです。LSWの逆コンパイラで.NETの中身を好き勝手に眺めては時間潰し。Javaと言い、.NETと言い、こんなにスケスケでいいのかなと思います。

Newの重み

JLCAで変換した私のJavaプログラムのファイルの日付は1999年になっています。そのファイルを見ると、Borland C++ Builder 3、 JDK 1.2、 TowerJ、Visual C++ 6.0、Visual J++ 6.0などを使っていた頃なのですね。ここではTower Jがネイティブなコンパイラです。これは、Javaのバイトコードを馬鹿正直にすべてCに置き換えてコンパイルするものでした。この会社は、もともとEiffelのコンパイラを作っていたのですが、途中でJavaに乗り換えました。つまり、Eiffelのコンパイラ技術をJavaに適用したのですが、惜しいことに昨年末に消え去ったもよう。ここなら問題なく.NETのネイティブなコンパイラを作れたであろうに、非常に残念です。しかし、ネイティブな.NETのコンパイラをGoogleで探すと、文字通り「NATIVE」（http://sourceforge.net/projects/native/）というのが引っかかりました。

下り坂のJavaとは対照的に、これからの.NETはますます充実してゆくのです！ そんな時期に、みなさんにとって一番重要なことは、プログラム言語のことは気にしないという態度です。重要なのは、ライブラリです。正確に言うならば、フレームワークです。言語なんて、フレームワークを機能させるための道具に過ぎません。材料はみなフレームワークのほうにあるのです。

さて、この頃、Javaに懐疑心をいだきながらも、私は結構真面目にJavaを使っていたように思えます。数値計算のライブラリにも凝りました。そして、実行時に作業用変数をNewする遅さに悲鳴をあげたものです。つまり、ある関数の中でローカルに用いる作業変数が、何らかのクラスのオブジェクトであるとすれば、それは「関数に飛び込んだときに動的にNewで生成されて、抜けるときは自動的に消滅する」ことになります。たとえば、次の関数をご覧ください。

```
void foo(Objekt x)
{
  Objekt t;
  …tを用いた計算…
}
```

ここでは、Objekt型変数tが作業変数ですが、C/C++の場合はこれを単に「Objekt t;」で済ますことができます。この場合には、作業変数は動的にヒープに生成されるのではなくて、スタック上に置かれます。C++であればObjektクラスの生成子は呼ばれますが、メモリ上での配置場所はコンパイル時に確保されているので、実行時にメモリマネージャからメモリをわけてもらう必要はありません。ところが、Javaではこうはゆきません。次のように「new」する必要があります。