bject BLVD オブジェクトの散歩道

例題でわかる！ .NET Framework

第10回 「.NETはオープンソース!?」
～.NETの逆コンパイラ～

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:

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。

SAMPO10.SLN
今回紹介したサンプルのソリューションファイル

移植性再考

Visual Basic .NETやC#は、プラットフォームを選ばないという移植性を売り物にしています。もっとも、ここで言うプラットフォームは、各種Windowsに限られていますが、とりあえずうるさいことは言わないことにします。とにかく、宿敵のJavaが移植性を売り物にしている以上、Microsoftも移植性にこだわらざるを得ません。今月はこの移植性なるものを、あえて話題にしたいと思います。

見出しを“再考”としたのは、Javaが流行り始めたころに、この移植性に関して結構議論を展開したので、それをご存知の読者の方もあろうかと思ってのことです。「移植性そのものの内容を問うことなく、それをスローガン化した怪しげなJava」に関する「移植性懐疑論」とでも呼ぶべき議論を展開したのは、前世紀の末でありました。しかし、まことに残念なことに、この議論は.NETの世界でも、ほとんどそのまま成り立ってしまうのです。このあたりの状況を今月はお話ししたいと思います。最初はプログラミングの話はありませんが、途中からちゃんとプログラミングが登場し、暗黒の中にも多少の光明を得るという段取りになっています。

Javaの移植性を考える

2種類ある移植性

何年も前に繰り返し私が論じたことなのですが「いわゆる“移植性”には2種類ある」という話から始めたいと思います。JavaCat.comというサイトに私の記事が載っていたので、それを引用しようと思って探したら、いつのまにか姿を消していました。パロアルトのはずれの高速道路脇にある日本の投資会社のサイトだったのですが、これもまた、アメリカでJavaが衰退している証拠のひとつかもしれません。仕方がないので、ここに要点を述べることにします。

Javaの移植性には2種類あります。

「実行時の移植性」と「開発時の移植性」です。実行時の移植性とは、ユーザーが各種プラットフォームで実行できるということです。開発時の移植性とは、開発者が各種プラットフォーム用に開発できるということです。この2種類の移植性を認識し区別することが非常に重要なのです。ところが、この2種類の区別は、意識して曖昧にされてきました。

世間一般向けの宣伝文句としては実行時の移植性を前面に出し、「PC用のワープロを買うと、Macでも使える」というようなフレーズまで登場しました。でも、このような製品が皆無であることはみなさんもご承知のとおりです。ここでは、Appletを除き「実行時の移植性は絵空事」であることを再確認したいと思います。

一方、開発時の移植性とは、開発者が各種プラットフォーム用のプログラムを、簡単に作成できることを指します。最近では、開発時の移植性はさらに充実しました。Java開発環境そのものは、過激な競争の結果、数種類にまで絞り込まれ、あまり選択の余地が残されていません。この過程は、技術的要因以外のものが大いに絡む“不自然淘汰”とも呼べるもので、私の大好きだったVisual Cafeも姿を消してしまいました。しかし、結果的には、Java開発環境の選択で失敗する余地はなくなったのですから、楽と言えば楽な話です。さらには、Java開発環境そのものまで、最近ではクロスプラットフォームになってきました。Javaの開発時の移植性には、目を見張るものがあります。

このように、開発時の移植性に関しては成功を収めたJavaですが、Javaそのものの現状は「UNIX用サーバー側ツール」という状況に陥ってしまいました。最近では、「携帯電話組み込み用ツール」として少々息を吹き返していますが、それでも「かなり特殊な開発者用」のツールであることには変わりありません。一般のアプリケーション開発者はついてきてくれませんし、広範なデスクトップ市場はWindowsに陵駕されたままです。これもみな、実行時のJavaに関してのツケが回ってきたからなのです。結果論になりますが、「実行時の移植性」なんて追及しなければよかったのです。それでは、この点に関して述べたいと思います。

実行時の移植性の代価

実行時の移植性に必要なものは、それぞれのプラットフォームにインストールされたJava実行環境です。つまり、Java仮想マシンとJava実行時ライブラリです。そして、Javaのプログラムは、Javaバイトコードの形で提供されます。そして、そのようなJavaアプリケーションプログラムの開発そのものには、技術的な問題があるようには思えません。Java開発環境はみな昔から、ブラウザなしで動くJavaアプリケーションプログラムの開発をサポートしています。それなのに、まったくと言って良いほど、Javaアプリケーションが存在しないのはどういうわけなのでしょうか。理由として考えられるのは、次のようなものです。

①設定が面倒
②軽快さに欠ける。重くて遅い
③守秘性に欠ける
④意識的に無視した

まず、①の設定が面倒とは、Java実行環境の設定が、結構面倒であるという事実です。誰にでもできるというものではありません。MicrosoftがWindowsに組み込まない限り、解決されない問題でしょう。そして、Windowsでの「DLL地獄」に相当する「ClassPath地獄」という言葉も存在します。これにネットワークが関係してくると、セキュリティまで絡んできて、話はさらに複雑になります。

②の軽快さに欠け重くて遅いとは、「Swing doesn't Swing」の一語に尽きます。Javaで作ったGUIは、例外なく軽快さを欠くのです。

そして、③の守秘性に欠けるというのが致命傷。バイトコードからJavaのソースコードを復元するのは、逆コンパイラ（decompiler）を用いれば実に簡単なのです。このような問題点の結果、Javaバイトコードにコンパイルされていても、「値打ちのあるものはユーザーには渡さない」ようになりました。値打ちのあること、すなわち「頭脳」に相当する部分はみなサーバーに任せ、いわゆるサーバーページと呼ばれる機能で「HTMLファイルを動的に生成して送り出し、クライアント側はそれをブラウザに表示するだけ」というパターンが確立してしまいました。

ここまでが、技術的な問題点①②③の結末です。ところが、それ以上に、ビジネスの観点から、実行時の移植性