επιστημηの

オブジェクト指向的夜話

C#から見た.NET Framework

新連載

C#とC++で.NETを覗き見してみる

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:
  Visual C++ .NET

Level

★★★☆☆

Samples

はじめに

これからしばらくご厄介になりますεπιστημηです。ご存知の方もいらっしゃるでしょうが、僕は根っからのC++屋です。C++とは産声を上げたころから、もう15年以上の付き合いでしょうか。産まれたころは言語仕様もシンプルで、ライブラリのほとんどがCの資産をそのまま使っていました。ここ最近（1995年あたり）、標準C++の制定により、C++はおそらく今日使われているプログラミング言語の中でも最も複雑怪奇なもののひとつになったように思います。C++に追加されていった新たな機能や仕様の変更は、産まれたころからずっとその成長を見守り続けた僕にとっては“ゆるゆる”と起こったことなのでさほどの違和感はありませんが、今からC++を始めようというビギナーには、700ページを優に越えるC++の言語仕様に恐れをなすに違いありません。C++は良くも悪くも「最高のプログラミング言語」となったのかと思います。

その傍らでJavaが台頭してきました。仮想機械上で実行されるJavaはマシン／OSの壁を越えて動き、ちょうど巻き起こったインターネットブームの波に見事に乗っかりました。そのタイミングの良さと共に、C++の難解さに辟易したプログラマの多くはJavaの言語仕様そのものの単純さ、そしてリッチなクラスライブラリに飛びつきました。なんといってもWebページの中でアプリケーションが動くってのが魅力的でしたね。今ではむしろサーバー側で活躍しているようですが。

そんな中、2002年3月にMicrosoft Visual Studio .NETが発売されます。新たな実行環境CLR（Common Language Runtime）、および.NET Frameworkという、.NET対

応言語であれば自由に呼び出せるライブラリが提供されました。そして.NET対応言語として新たに据えられたのがC#ですわね。

Visual Studio .NET発売以前にも.NETはとかく話題にはなっていました。C#とかいう新言語が発表されるらしいとか、JavaとC++のいいとこ取りだとか、そんな前評判（?）が業界を賑わせていましたが、僕はその頃（今でもですが）C++にベタ惚れだったのでさほど大きな興味も示さず、Visual Studio .NETを手に入れてみたものの、一番欲しかったのはその中に収められたVisual C++ Ver7“だけ”でして、C#、Visual Basic .NET、J#の類はインストールこそしましたがほとんど触らずじまいで2年の月日が流れてしまいました。僕のカラダがC++に馴染みすぎちゃっていて、いまさらわざわざ他の言語で苦労しなくても……、という思いがあったのは否めません。C++ならマニュアルなんかなくてもエディタ立ち上げていきなりコードを叩き込むことができるわけですから。

とはいいながら、Javaはドキュメント首っ引きとはいえそこそこ書けるんですよ。だからC#だってマニュアル引きつつぼちぼち手を染めてみようかと思い立ったという次第。幸いにも僕はオブジェクト指向とかデザインパターンとかについては心得がありますから、あとはアタマの中にあるアイデアをC#でどう表現するか、なんですよね。そんなわけで今回から始まる僕の連載、オブジェクト指向屋兼C++屋（そしてほんの少しJava屋）の僕がC#世界で見たこと聞いたこと試してみたことそして考えたことなどなど、筆（キーボード）にまかせて書き綴っていこうと考えています。

C#および.NET Framework ～First Impression

C++歴の長い僕からみた、C#そして.NET Framework（ライブラリ）の印象を思いつくまま書き並べてみます。中には無知からくる誤解もありましょうが……。

○ファイル構成がシンプル

C++では通常宣言と実装をそれぞれ別のファイルに記述します。一般に宣言部（ヘッダ）は拡張子として“.h”、実装部は“.cpp”や“.cc”“.cxx”などが用いられます。

どこか他の場所から変数や関数を参照させるとき、その名前や型、呼び出し形式を公開しなければなりません。そのために宣言部を分離してヘッダとするわけですね。

例：hello.h

```
#ifndef HELLO_H__
#define HELLO_H__

class hello {
public:
  void say();
};

#endif
```

例：hello.cpp

```
#include <iostream>
#include "hello.h"

void hello::say() {
  std::cout << "Hello" << std::endl;
}
```

C#は（Javaも）このような分離を必要としません（というか、分離できません）。そのために、アプリケーションの構築に必要なソースファイルの数がC++に比べて少なくなります。

例：hello.cs

```
class hello {
  public void say() {
    System.Console.WriteLine("Hello");
  }
}
```

宣言と実装を分離する言語は最近はめっきり少なくなってきました。管理が面倒なんですよね。ヘッダに変更が加わるとそのヘッダに依存する（直接／間接的に参照している）すべての実装をコンパイルし直しとなります。それはC#だろうがJavaだろうが同じなのですが、C++の場合依存関係の判定はプログラマの責任なので、うっかり再コンパイルを忘れたりします。ソースコードそのものがシンプルになるわけで、C++屋には羨ましい仕様です。とはいえ、宣言と実装を分離するのも時と場合によ