世界はオブジェクトの海に浮かぶ

.NET Framework で楽しむ オブジェクト指向

最終回 STL/CLR

επιστημη
えぴすてーめー

はじめに

今回で最後ですって。心残りではあるけれど……。この連載、C#とC++/CLIネタをメインにいろんなトピックを取り上げてきました。僕の書くのは（昔からなんだけど）取り立てて見栄えのする新機能とかではなく、日頃書いてるコードからヒントを得たものや昔書いたコードを今風にリライトしたものが多かったように思います。連載の最後くらいは“近未来のお話”で締めくくるとしましょかね。

STL.NETをご存知ですか？

Visual Studio 2005のリリースに先立って発表された「Microsoft 開発ツール ロードマップ」にVisual Studio 2005の新機能として以下の一節があります。

「STL（Standard Template Library）の新バージョンが導入されます。このバージョンのSTLは、マネージコードやデータと相互作用するようにチューニングされています。従来のC++アプリケーションの作成にSTLを使い慣れているプログラマは、CLRベースのアプリケーションの作成にも同じ技法を適用できることがわかります。」

僕はSTLとは浅からぬ付き合いなのでこのロードマップを読んだときに「お、なんか面白いことを仕組んでくれそうだぞ」と心踊らされました。

その後Microsoft Visual C++チームのアーキテクトStanley B. Lippman氏によるアーティクル「STL.NET Primer」（邦訳「STL.NET 入門」）が2004年8月に発表されています。

このアーティクルの冒頭に、

「Visual C++ 2005では、標準テンプレートライブラリ（STL）が.NET Frameworkで使用できるように再設計されています。シリーズの第1回となる今回は、STL.NETの概要を紹介します。」

とありました。そうか、コレがロードマップにあったSTLの新バージョンってやつか。

ここで少々長いのですが引用させてください。邦訳「STL.NET 入門」ではSTL.NETのメリットを次ページの囲みのように述べています。

C++プログラマにはお馴染みのSTLが提供するコンテナ（オブジェクトの集合）は、要素の格納に必要なメ

レベル >>> Level

1 2 3 4 5

言語 >>> Language

- C#
- C++/CLI

ツール >>> Tool

- Visual Studio 2005 Professional
- Virtual PC
- Visual Studio Code Name "Orcas" - September Community Technology Preview

MSDN「STL.NET 入門」より

STL.NETの詳細に入る前に、ある意味避けては通れない疑問があります。Visual C++プログラマが、System::CollectionsやSystem::Collections::Generic などの言語に依存しないライブラリの代わりにSTL.NETコンテナライブラリを使用するメリットはどこにあるのでしょうか。
一般に、System::Collectionsライブラリはまっ先に選択肢から外れます。その理由は、Visual Studio 2005でGenericライブラリが用意されることになった理由と同じです。すなわち、パラメータ化のオブジェクトモデルが複雑であり、型情報が失われるため安全でないからです。単純な用途については問題ありません。これは、要素数が16以下のコンテナではバブルソートを使用しても問題にならないのと同じです。しかし、アプリケーションで現実的な問題に対処するには、より洗練されたソリューションが必要になります。
したがって、Visual C++のようなシステムプログラミング言語にとっての選択肢は、STL.NETとSystem::Collections::Genericライブラリのいずれかになります。では、Visual C++プログラマがこの2つのうちSTL.NETを選択する理由はどこにあるのでしょうか。また、STL.NETを選択するとして、プログラムが他の.NET言語から孤立する心配はないのでしょうか。これらは当然の疑問であり、ここで取り上げる価値があります。
1つ目の答えは、拡張性です。Alex Stepanov氏によって考案されたSTLのオリジナルデザインパターンでは、アルゴリズムとコンテナが別のドメイン空間に分離されています。したがって、すべてのコンテナに適用できるアルゴリズムを追加したり、アルゴリズムを適用できるコンテナを追加したりできます。これに比べると、Genericライブラリは従来のコンテナモデルに近いと言えます。このことは、2つ目の答えにもつながります。
2つ目の答えは、統一性です。現役のC++プログラマがこれまでに築き上げてきたものは、既存のコードばかりではありません。STLライブラリを使ったプログラミングのノウハウも大事な資産です。私たちは、既存のコードだけでなくそうしたノウハウも移行できるようにしたいと考えています。それまでSTLを利用してきたC++プログラマにとって（STLを利用していない現役のC++プログラマはまずいないでしょう）、.NETでそれを利用できないことは大きなマイナスです。少なくとも、筆者はそう感じてきました。これについては、多くの熟練C++プログラマから懸念の声が上がっており、彼らが.NET への移行を見合わせる理由となっています。
3つ目の答えは、パフォーマンスです。しかし、C++プログラマはパフォーマンスの問題にうるさいことで有名なので、ここでこの問題に踏み込むのはやめて、後の記事で取り上げることにします。最後に、STL.NETは確かにすばらしいが、それを使用することによってC++プログラマやC++/CLIプログラムが.NETコミュニティから孤立するのではないか、という疑問にお答えしておきましょう。その答えはノーです。私はそう思っています。この問題については、Anson Tsao、Martyn Lovell、P.J. Plaugerを始めとするSTL.NETのアーキテクトが徹底的な調査を行っており、IEnumerator、IList、およびICollectionのサポートを通じて他の.NET言語との互換性を確保できるようになっています。

モリの管理をtemplate引数に与えるallocatorを使って自前で行ないます（とはいえこのallocatorは“まず間違いなく”省略されるので意識されることがないのですが）。ところが.NET Frameworkで用いられるManagedオブジェクトはメモリ管理を.NETに委ねるので、STLコンテナの要素にManagedオブジェクトを格納することができません。

```
#include <vector>
std::vector<System::String^> vec;  // エラー
```

「STL.NET 入門」によると、新たな名前空間cliにSTLと同じvector、list、setなどManagedオブジェクトを要素とするコンテナを用意してくれるとのこと。.NET Frameworkには名前空間System::Collections::Genericにさまざまなコンテナを用意してくれていますが、C++に首まで浸かった僕にはSTLのほうが馴染みがよろしい。なのでSTL.NETをわくわくしながら待ち望みました。

ところがどうも様子がおかしい。「STL.NET 入門」は連載のはずなのに第2回以降が出る気配もない。どうなっちゃったんだろ、Visual C++ 2005のリリースに間に合うのかしら？　とヤキモキしながらMicrosoft開発チームのBlogを見守っておりました。結局昨年暮れのVisual Studio 2005リリース時にはSTL.NETは提供されず、その後のサービスパックまで“おあずけ”とのこと。さらにその後STL.NETはSTL/CLRと名前を改め、大幅に遅れて次期Visual Studio、コードネーム“Orcas”に搭載との情報を得ました。

なんだかなー、この調子じゃまたズルズルと遅れて結局Orcasにも入らないんじゃないかと半ばあきらめムードの漂っていた頃、Visual C++ Team Blog（http://blogs.