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

第4回 generics対応のコレクションとメソッド

επιστημη
えぴすてーめー

Level

1 2 **3** 4 5

Technology Tools

- ☐ Visual Basic
- ☑ Visual C#
- ☐ Visual C++
- ☐ SQL Server
- ☐ Oracle
- ☐ Access
- ☐ ASP.NET
- ☐ Other:
  Visual Studio 2005 ベータ1

Samples

Partial classの恩恵

Visual Studio 2005のリリースがじわじわと近づいてきました。先月号のオマケ「Visual Studio 2005 β2日本語版」をインストールした方も数多くいらっしゃることでしょう。僕は思うところあって、まだβ1のままです。VS2005はVS2003に比べて、より洗練された感じがします。僕のホームグラウンドはC++でして、IDEを立ち上げても使うコンパイラは大抵C++かC#、アルゴリズムをコネ回すのが大好きなうえ“絵心”ってもんが情けないほどに欠如しているものだから、GUIアプリケーションをこしらえる機会が極めて少ないのです。もっぱらコマンドプロンプトからコマンドラインコンパイラ「cl/csc」でコンパイルってスタイルです。先日初めてVS 2005 β1でC# GUIお遊びアプリケーションを書きました。コードがとっても涼しくなってますね。C#2.0で導入された機能「Partial class」によってクラスの実装を複数のソースに分割でき、IDEはこれを使ってフォームデザイナが吐いたコード（図1・リスト1）と僕らプログラマが手入力するコード（リスト2）とがきちんと分離されます。たとえば従来プライマリフォームのコードがForm1.cs一本であったのが、デザイナの吐くコードはForm1.Designer.cs、僕の書いたコードはForm1.csとなり、Form1.csには“ここはデザイナが生成した部分だから触わらないで！”な部分が混じりません。さらにデザイナ側でフォームに貼り付けたコントロールの名前を変更するとForm1.csにある名前もきちんと置き換えてくれますし、リファクタリング機能を使えばForm1.cs側でコントロール名を変更することも可能です。C#2.0で追加されたさまざまな言語仕様の拡張にIDEが追随してくれています。これは実

リスト1：フォームデザイナが吐いたコード

```
namespace simple
{
   partial class Form1
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.textBox1 = new System.Windows.Forms.TextBox();
         this.button1 = new System.Windows.Forms.Button();
         this.SuspendLayout();
//
// textBox1
//
         this.textBox1.Location = new System.Drawing.Point(19, 24);
         this.textBox1.Name = "textBox1";
         this.textBox1.Size = new System.Drawing.Size(126, 19);
         this.textBox1.TabIndex = 0;
//
// button1
//
         this.button1.Location = new System.Drawing.Point(168, 26);
         this.button1.Name = "button1";
         this.button1.Size = new System.Drawing.Size(68, 16);
         this.button1.TabIndex = 1;
         this.button1.Text = "button1";
         this.button1.Click +=
           new System.EventHandler(this.button1_Click);
//
// Form1
//
         this.AutoScaleBaseSize = new System.Drawing.Size(5, 12);
         this.ClientSize = new System.Drawing.Size(292, 273);
         this.Controls.Add(this.button1);
         this.Controls.Add(this.textBox1);
         this.Name = "Form1";
         this.Text = "Form1";
         this.ResumeLayout(false);
         this.PerformLayout();
      }

      #endregion

      private System.Windows.Forms.TextBox textBox1;
      private System.Windows.Forms.Button button1;
   }
}
```

に気持ちがいい。機械がまた一歩、人間に歩み寄ってくれた気がします。

genericsと.NET Framework

VS2005ではC#と.NET Frameworkのバージョンが2.0となります。それにより、C++とJavaで実現されていたgenerics（ジェネリックス）がC#（およびVB.NET）でも使えるようになりました。genericsについては先月号の特集記事「Visual C# 2005に見る新機能」でこだかさんが解説してくださってますのでそちらを参照してくださいませ。genericsはC++屋の僕には待ちに待った機能です。ひとつのコードで何にでも使える万能（とまではいかないけれど）クラスや万能メソッドの実装が可能になりますからね。

図1：VS2005 β1で作ったC# GUIお遊びアプリケーション

言語仕様としてのgenericsはいいとして、.NET Frameworkはgenericsに対応したライブラリを提供してくれているのでしょうか。β1に付いてくるドキュメント（MSDN）とオブジェクトブラウザで調べてみました。