実践！次世代 C/S システム

クライアント/サーバー

スマートクライアント＆Webサービスの実際

中垣 健志
NAKAGAKI, Kenji
株式会社CSKシステムズ
IT生産技術部

最終回

バージョン管理とアプリケーションの配布

バージョンと配布

.NET Framework 2.0が正式にリリースされてから一年と数か月が経ちましたが、早くもマイクロソフトは.NET Framework 3.0の情報を公開しています。しかし.NET Frameworkのようなすべてのアプリケーションの基盤となるフレームワークが頻繁にリリースされると利用者の混乱を招くため、万全の足元固めをしたうえで提供してもらいたいものです。

かたや企業で利用されるエンタープライズアプリケーションを見てみると、時間を十分にかけたリリース計画を立てる余裕はないといっていいでしょう。ビジネスルールは企業の都合を無視して、どんどん新しくなっていきます。このようなビジネスルールの変化に対して迅速に対応しなければ、アプリケーションの価値はどんどん低くなっていきます。またビジネスルールの変化に終わりがないように、アプリケーションの開発も「これで終わり」という状態は存在しません。常に改修を加えて形を変えながら、長いライフサイクルをたどっていきます。

このようなアプリケーション開発では実装にかかるコストはもちろんのこと、バージョン管理とアプリケーション配布に関わるコストも十分考慮する必要があります。バージョン管理がしっかり行なわれていなければ、不具合対応や改修を行なう必要があるときにどのソースコードを修正すればいいのかわかりません。また場合によっては何百何千台と存在するクライアントに対するアプリケーションの配布コストを低く見積もってしまうと、システム管理者はもとより、アプリケーションを利用する何百何千というユーザーの時間を無駄に消費させてしまいます。

.NET Frameworkではこのような問題を解決するために、アセンブリにバージョン管理の仕組みを組み込んだり、ClickOnceといった配布技術の提供を行なっています。しかしこれらの技術はあくまで手段であり、適

レベル >>> Level

❶ ❷ ❸ ❹ ⑤

言語 >>> Language

- Visual Basic
- C#

ツール >>> Tool

- Visual Studio 2005 Professional
- SQL Server 2005

サンプル >>> Sample

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、
http://www.shoeisha.com/mag/windev/
からダウンロード可能です。

切に使用できなければ効果を上げることはできません。

そこで今回はこれらの技術にスポットを当てて、効率的にバージョン管理やアプリケーションの配布を行なうためのノウハウを紹介していきます。

アセンブリのバージョン番号の仕組み

.NET Frameworkで作成されたプログラムは「アセンブリ」という形にまとめられます。このアセンブリはEXE（.exe）またはDLL（.dll）というファイル形式で提供されており、必ずバージョン番号を持っています。アセンブリのバージョン番号は、ファイルのプロパティを開くことで確認できます（図1）。ではこのバージョン番号の違いは、アプリケーションの挙動にどのような影響を及ぼすのでしょうか？

実はアセンブリのバージョンの違いは、あるアセンブリから別のアセンブリを利用するときに影響があります。ここではこの影響を確認するために、次の2つのアセンブリを用意します。

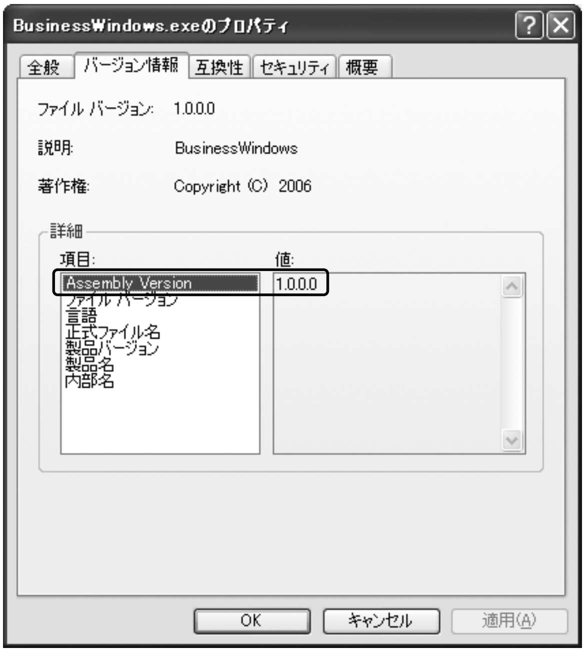
①コンソールアプリケーションとしてビルドした「A.exe」
②クラスライブラリアプリケーションとしてビルドした「B.dll」

2つのアセンブリのバージョンはそれぞれ、「1.0.0.0」となっています。そして「A.exe」は「B.dll」を参照しています（リスト1）。

このプログラムを実行すると、図2のようにそれぞれのアセンブリ情報が表示されます。

この場合、「A.exe」が個々のアプリケーション、「B.dll」がさまざまなプロジェクトで利用されている共通部

図1：アセンブリのバージョン番号の確認



BusinessWindows.exeのプロパティ

ファイル バージョン: 1.0.0.0
説明: BusinessWindows
著作権: Copyright (C) 2006

詳細

項目:	値:
Assembly Version	1.0.0.0
ファイル バージョン	
言語	
正式ファイル名	
製品バージョン	
製品名	
内部名	

リスト1：あるアセンブリから別のアセンブリを利用する（その1）

Visual Basic

```
' A.exe（コンソールアプリケーション）
Module Module1
   Sub Main()
      Console.WriteLine(GetType(Module1).Assembly.FullName)
      Class1.ShowAssemblyInfo()
   End Sub
End Module

' B.dll（クラスライブラリ）
Public Class Class1
   Public Shared Sub ShowAssemblyInfo()
      Console.WriteLine(GetType(Class1).Assembly.FullName)
   End Sub
End Class
```

C#

```
// A.exe（コンソールアプリケーション）
class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine(typeof(Program).Assembly.FullName);
      Class1.ShowAssemblyInfo();
   }
}

// B.dll（クラスライブラリ）
public class Class1
{
   public static void ShowAssemblyInfo()
   {
      Console.WriteLine(typeof(Class1).Assembly.FullName);
   }
}
```