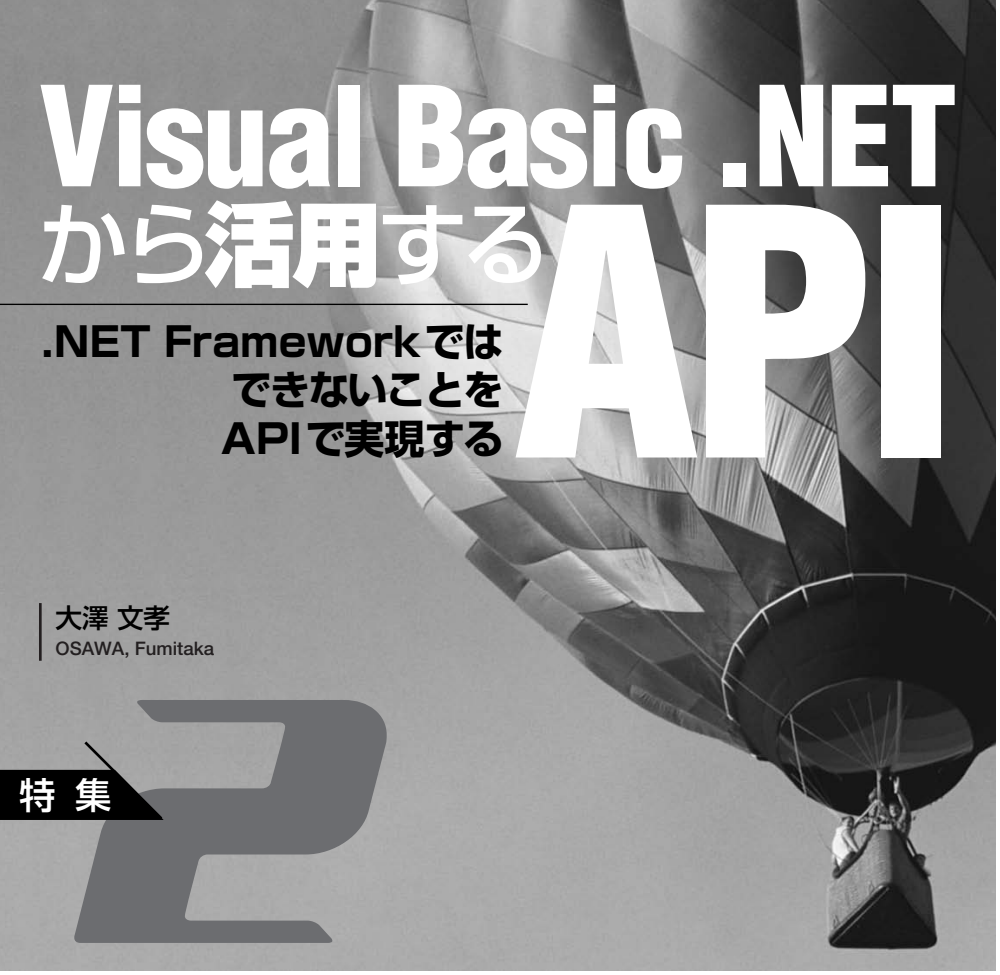

Visual Basic .NETから活用するAPI

.NET Frameworkではできないことを APIで実現する

大澤 文孝
OSAWA, Fumitaka

特集2

はじめに

.NET Frameworkクラスライブラリには、豊富な機能が備わっています。しかしWindowsのすべての機能を使えるわけではありません。

ハードウェア寄りの機能が使えないのはもちろんですが、なかには「なんでこんな簡単なことができないのか」と思う場面もあります。

そんなときには愚痴を言っても仕方ないので、COMを使ったりWin32 APIを使ったりして、.NET Frameworkから外に出る必要があります。

.NET Frameworkの外に出るための機能は、「P/Invoke」と呼ばれます。

本稿では、P/Invokeを使って、COMやWin32 APIを使う方法を説明します。

Level

1 2 3 4 5

Technology Tools

- ✓ Visual Basic
- ○ Visual C#
- ○ Visual C++
- ○ SQL Server
- ○ Oracle
- ○ Access
- ○ ASP.NET
- ✓ Other:
  - ↓
  - Win32 API
  - Visual Studio .NET 2003
  - Visual Studio 2005

P/Invokeのセキュリティ制限

具体的な使い方を説明する前に、P/Invokeを使うときの注意点を述べておきます。

P/Invokeを使うと、Win32バイナリを実行できるため、いわば、「何でもあり」の状態になります。

そのためP/Invokeを使うときには、次の点に注意が必要です。

・動作が不安定になる可能性がある

呼び出し方を間違えると、メモリリークが生じたり、動作が不安定になったり、最悪の場合には、OSの保護違反が発生して強制終了させられる可能性もあります。

・実行にはUnmanagedCode権限が必要

.NET Frameworkには、メソッドやプロパティ単位で、どのような権限が必要となるのかを検証し、権限がないときにはセキュリティ例外を発生する機構があります。この機構によってユーザーは、安全ではない動作をするコードの実行から身を守ることができます。

Win32バイナリを実行すれば、安全性が失われるのは明らかです。

そこで.NET Frameworkでは、「UnmanagedCode」というセキュリティ権限がないと、P/Invokeを使えないように保護する構成がとられています[注1]。

UnmanagedCodeセキュリティ権限がない場合には、図1のように例外が発生します。

デフォルトでは、ローカルから実行されるコードにだけ、UnmanagedCode

注1）.NET Framework 2.0 SDK（Visual Studio 2005）には、「permcalc.exe」というセキュリティ権限を調べるコマンドラインツールが付属しています。このツールを使うと、そのアセンブリを実行するのに、どのような権限が必要なのかがわかります。

図1：必要な権限がないときに発生する例外

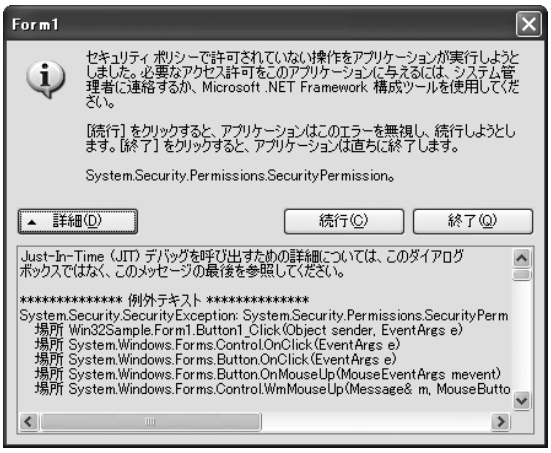

図2：COMコンポーネントを参照する

セキュリティ権限が付いています[注2]。

・OSの違いによる動作検証が必要になることがある

.NET Frameworkは、Windows 9x系でも、Windows NT系でも、はたまた最近登場した64ビットWindows XPでも動きますが、Win32 API（Win64 API）は、OSに固有のものになります。

そのためどのOSでも正しく動くことを求めるなら、きちんとした検証作業が必要になります。

これらの3つの注意点を総合的に勘案すると、

必要なければ使わないほうがよい

ということになります。

COMやWin32 APIを使う前に、まず、.NET Frameworkのライブラリで実現不可能なのかどうかを確認してください。

そしてどうしても実現不可能であったときに限って、COMやWin32 APIを使うようにすることを推奨します。

COMを使う

まずは、COMの使い方から説明します。

.NET FrameworkからCOMを使うには、参照設定するだけなので、見かけは簡単です。しかしそれでも、いくつか注意しなければならない事項があります。

COMの参照設定

Visual Studio 2005から、「プロジェクト」メニューの「参照の追加」を選ぶと、COMコンポーネントを参照設定できます[注3]。

たとえば図2のように「Microsoft Scripting Runtime」を参照設定した場合には、Scripting名前空間に、このCOMコンポーネントが含まれるようになります[注4][注5]。

このときたとえば次のようにすると、このCOMコンポーネント内の「FileSystemObject」クラスのインスタンスを作れます[注6]。

```
Dim scripting As New Scripting.FileSystemObject()
```

インスタンスを作ったならば、

```
Dim folder As Scripting.Folder = _
 scripting.GetSpecialFolder( _
  Global.Scripting.SpecialFolderConst.SystemFolder)
MsgBox(folder.Path)
```

注2）セキュリティポリシーは、コントロールパネルの管理ツールにある「Microsoft .NET Framework構成ツール」で変更できます。
注3）本稿ではVisual Studio 2005を使って説明していきますが、Visual Studio .NETやVisual Studio .NET 2003でも同じ操作ができます。
注4）Microsoft Scripting RuntimeというCOMコンポーネントは、VBScript（Windows Scripting Host）でファイル操作などをするときに使われているCOMコンポーネントです。
注5）どのような名前空間で取り込まれるのかは、COMコンポーネントによって異なります。
注6）ここではCOMを使っているということをわかりやすくするために、あえて「Scripting.FileSystemObject」とフルネームで記述しています。しかし「Imports Scripting」としておけば、「FileSystemObject」のように表記を省略できます。