SKILL UP!

将来にわたって使えるスキルを身につけるために

オブジェクト指向ではじめるプログラミング

日向 俊二
HYUGA, Shunji

第5回

ウィンドウアプリケーションを作る（その1）

Technology Tools

- ❑ Visual Basic .NET
- ☑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ☑ Other:
  .NET Framework SDK

Level

★ ★ ★ ★ ★

Samples

・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、付録CD-ROMの¥DOTNET¥CSHARPディレクトリに収録しています。

¥AWINDOW
単純なウィンドウ表示プログラム

¥HELLOWND
ウィンドウ版Hello C#プログラム

※この記事では.NET Framework SDKを利用して解説していますが、C#Builder、SharpDevelopでも利用できます（C#コンパイラならすべて利用できます）。

はじめに

C#は、純粋なプログラミング言語です。プログラミング言語としてのC#には、ウィンドウもボタンもツールバーも含まれていません。これまでの4回の連載で取り上げたサンプルプログラムも、ウィンドウを使わない“コンソールアプリケーション”だけでした。だからといって、C#でWindowsのウィンドウアプリケーションやGUIプログラムを開発できないというわけではありません。C#と.NET Framework SDKだけでも、ウィンドウアプリケーションやUIを備えたコンポーネントを開発することができます。

C#とWindowsアプリケーションの開発の関係は、C/C++とWindowsアプリケーションの開発の関係に似ています。C/C++もプログラミング言語ですから、ウィンドウもボタンも含まれていません。そのため、C/C++でウィンドウを利用するアプリケーションを開発するためには、WindowsのAPIを直接呼び出したり、MFCのようなWindowsのGUIをサポートするクラスライブラリを使う必要があります。しかし、OS開発のための言語として出発してオブジェクト指向プログラミングに利用されるようになった「C/C++を使うWindowsプログラミング」と、最初からWindowsの開発をターゲットとして開発された「C#を使ったGUIプログラム開発」との間には、決定的な違いがあります。

C#には、Windowsアプリケーションやコンポーネントの開発が容易になる“タネ”と“シカケ”があるという点です。

今回と次回の2回で、C#と.NET Framework SDKを使って、Windowsアプリケーションを開発してみましょう

C#と.NET Framework

C#は、ECMA（European Computer Manufacturer's Association、欧州コンピュータ製造業者協会）の標準として仕様が公開されているプログラミング言語です。つまり、プログラミング言

★1　現在の一般的なウィンドウシステムでは、ほとんどこの種のメッセージループを使ってユーザーやシステムとの対話を制御し管理します。ただし、プログラミング言語や開発環境の中にはメッセージループをプログラマに見せないように背後に隠蔽しているものがあります。また、きわめて特殊な構造のプログラムの中には例外的にメッセージループを使わないものもあります。

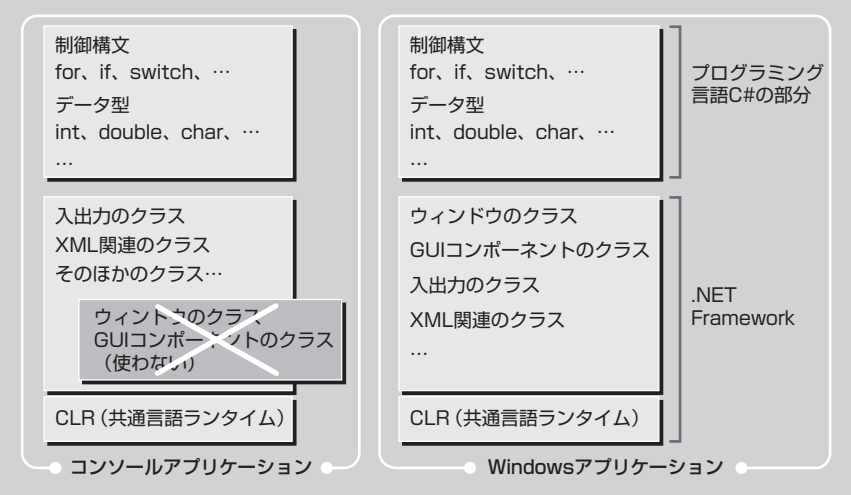
語としてのC#は、WindowsやNET Frameworkとは直接関係がありません。これは、Visual Basicを代表とするこれまでのMicrosoftの主要なプログラミング環境とは大きく異なる点であり、C#がユニークである理由でもあります。

しかし、もちろん、C#はその開発当初から、「Windows」という実行環境をターゲットとしていました。

そして、この連載のコンソールアプリケーションのサンプルの開発にも、C#のオブジェクト指向のアプローチと、.NET Frameworkを使ってきました。実際、これまでのサンプルでは、.NET Frameworkのうちの、CLR（Common Language Runtime）と、クラスライブラリの一部（GUIを除いた部分）を使っていました（図1左）。

GUIを活用する（ウィンドウを表示する）Windowsアプリケーションを作るために必要なことも、これまでに説明したこととほとんど同じです。ただひとつ違う点は、「.NET Frameworkの一連のクラスライブラリの中の“GUI関連のクラス”も使う」ということだけです（図1右）。

図1：コンソールアプリケーションとWindowsアプリケーション



*）ここでは例としてアプリケーション開発の場合で説明していますが、コンポーネントの開発でも考え方は同じです。

ウィンドウを作るには？

さて、それではさっそくC#と.NET Framework SDKを使って、Windowsのウィンドウを作ってみましょう。

.NET Frameworkでは、通常、ウィンドウは「Formsクラスの派生クラス」として定義します。つまり、単純なウィンドウのクラスを定義したいときには、次のようにすればよいわけです。

```
public class Form1 :
System.Windows.Forms.Form
{
}
```

これで「Form1」という名前のフォームのクラス、つまり、実行時にはウィンドウとして表示されるはずのクラスができたわけですが、これだけではC#のプログラムとして実行することはできません。

単独で起動されるC#のプログラムは、「Main()」からプログラムの実行が開始されます。つまり、C#の起動可能なプログラムには、次のようなプログラムの開始点（アプリケーションのメインエントリポイント）が必要です。

```
static void Main()
{
 // アプリケーションのコード
}
```

「このMain()からプログラムが始まる」ということは、この連載で前回までに取り上げてきたコンソールアプリケーションでも、ウィンドウを使うアプリケーションでも同じです。

◆Main() で実行するコード

さて、ここで実行するべき「アプリケーションのコード」について考えてみましょう。

フォームベースのアプリケーション、つまり、ウィンドウを使うアプリケーションは、何らかのメッセージを受け取らないと、ユーザーと対話できないだけでなく、ユーザーの通常の操作でプログラムを終了することさえできません。ですから、開発に使うプログラミング言語が何であろうと（また、Windowsであろうとほかのウィンドウシステムであろうと）、ウィンドウを使うアプリケーションには「ウィンドウに送られてくるメッセージを処理するループ」が必要です★1。

このループは、通常は、プログラムが終了するまで“メッセージを受け取っては処理する”ということを繰り返すだけなので、「メッセージループ」と呼びます。