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

第3回

プレゼンテーションレイヤーを作成する

ブラウザからスマートクライアントへ

最近WebアプリケーションではAjaxが脚光を浴びています。GoogleMapに代表されるこの仕組みがもてはやされるということは、既存の不自由なブラウザベースのインターフェイスに不満を持つユーザーがいかに多かったかということを示しています。しかし業務アプリケーションの世界においては、もっともっとリッチでスマートなインターフェイスが要求されます。

イントラネットでOSをWindowsに限定してもよいという条件がつくならば、ブラウザアプリケーションからスマートクライアントへの移行を真剣に考えるべきでしょう。なぜならばユーザーはシステムに対して、いかに早く効率的に仕事をこなしてくれるかという点を重要視するからです。

Windowsフォームを作成するに当たって

コンピュータが登場した黎明期からしばらくの間、ユーザーはコンピュータを操作するためにキーボードからすべての命令を入力する必要がありました。しかしGUIという新しいインターフェイスが発明[注1]されてからは、一般の利用者はキーボードと同じ程度にマウスをよく使うようになりました。Windowsは（その名前が表わすように）最初から、このGUIとマウスを使ったインターフェイスを使うことができたOSのひとつです。そして.NET以前のVisual Basic時代から、開発者は使いやすいWindowsフォームを簡単に実装できる開発環境を手に入れていました。

GUIとマウスというインターフェイスは、コンピュータを初めて使う人にとっては直感的に利用できます。しかしエンタープライズシステムでは、わかりやすいこともさることながら「定型作業においてデータをすばやく入力できる」ことも非常に重要な要

レベル >>> Level

1 2 3 4 5

言語 >>> Language

- Visual Basic
- C#

ツール >>> Tool

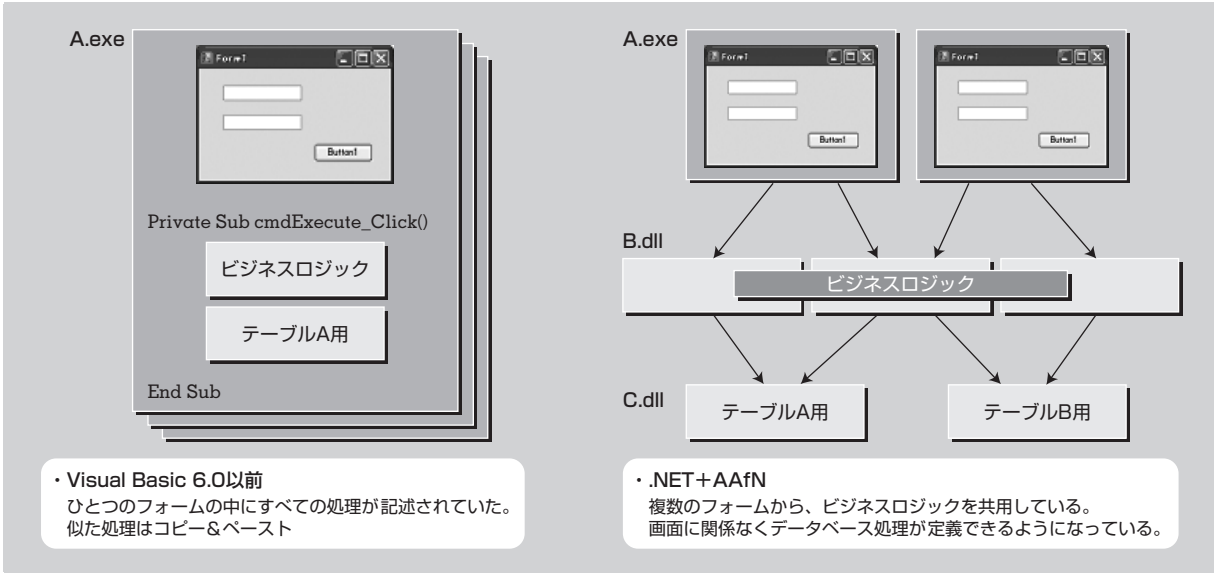
- Visual Studio 2005 Professional
- SQL Server 2005

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。

注1）歴史的にはLisaが、一般ユーザーがGUIとマウスを手に入れることができた最初のコンピュータだといわれています。

図1：Visual Basic 6.0以前と.NET＋AAfNの比較



素です。この2つが合わさって初めて、利用者にとって役に立つITの仕組みと言えるでしょう。

※プレゼンテーションレイヤーで実装するべきこと

Windowsフォーム上にはボタンやテキストボックスが配置され、ユーザーからの入力を待機する状態になっています。ユーザーはマウスやキーボードを使って、データを入力したり処理を実行したりします。この順序は必ずしも固定である必要はなく、好きな順番でデータの入力や処理の実行を行なうことができます。

しかしプログラミング言語という視点で見てみると、通常は上から順番にプログラムが実行されることになっています。このため単にBASICやC言語の実装方法ではGUIのプログラムを記述することはできません。そこでVisual Basicでは「イベント駆動型」のプログラミングスタイルを開発者に提供しました。イベント駆動型のプログラムでは、「ボタンが押された」「文字が入力された」というようなイベントに対し、そのイベントに応じた処理を記述します。これにより開発者は、簡単にGUIのプログラムを記述することができるようになりました。

さて、もう少し詳しくこのイベント駆動型のプログラミングについて見てみましょう。エンタープライズアプリケーションでボタンが押されたときは、通常は何かしらの業務処理を実行しようとしているときです。この処理では、画面上の情報の取得や情報の内容による処理の分岐、データベースの操作が行なわれます。.NET以前のVisual Basicでは、これらの処理がまとめてひとつのフォーム内に実装されていることがよくありました。しかしそのような実装を行なった場合には、処理が不可分なものとなってしまいます。たとえば実際のアプリケーション開発では、同じテーブルを操作する複数の処理があるかもしれません。また、同じ処理を行なうために、WindowsフォームとPDAなど複数のユーザーインターフェイスを用意するかもしれません。そのようなことも考慮して、.NET FrameworkではApplication Architecture for .NET（以下AAfN）というアーキテクチャに従って、複数のレイヤーでひとつのイベントを実装することを勧めています。

つまり、プレゼンテーションレイヤーに含まれるイベントプロシージャでは画面に関する必要最低限の処理のみを記述して、本質的な業務ロジックやデータベースアクセスのプログラムはバックエンドのレイヤーに任せることが必要となってきます（図1）。

ビジネスロジックとテーブル操作用のクラスは次回以降で解説することにして、ここではプレゼンテーションレイヤーに実装する画面処理について、さらに詳しく見