アドバンスド Windows プログラミング

Microsoftテクノロジーをめぐる冒険

第9回

ClickOnce

こだか かおる
KODAKA, Kaoru

はじめに

ついにVisual Studio 2005／SQL Server 2005の提供が開始されました。といっても、この原稿を執筆している時点では、まだ日本語版の提供は開始されておらず、MSDN Subscriber Downloadで英語版が入手できるのみです。

せっかく製品版が利用できるのですから、わざわざベータ2やRCを使うこともありません。ということで、今回の記事では、Visual Studio 2005 Professional Edition英語版を利用しています。そのため、表記などが英語になってしまっていますが、適宜、日本語に読み替えてください。

Level

1 2 3 4 5

Technology Tools

- ✓ Visual Basic
- ○ Visual C#
- ○ Visual C++
- ○ SQL Server
- ○ Oracle
- ○ Access
- ○ ASP.NET
- ✓ Other:
  - ↓
  - Visual Studio 2005 Professional Edition英語版
  - Apache
  - Mozilla Firefox

アプリケーションモデルの変遷

今から10年くらい前、「オフコンやメインフレームからのダウンサイジング」が叫ばれていた頃は、クライアント／サーバー型のアプリケーションが主流でした。ちょうどその頃、Windows 95が登場し、それまでの開発ツールに比べ、格段にアプリケーションの開発がしやすくなったVisual Basicがもてはやされます。その後、インターネット時代の到来とともに、クライアント／サーバー型からWebアプリケーションへと主流が移り変わっていきました。

MicrosoftもIIS、ASPを投入することで、Visual Basicを利用していた開発者をASPの世界に取り込み、一気にクライアント／サーバー型を駆逐するかに思われました。しかし、Webアプリケーションのデメリット、すなわち「表現力が乏しい」という点はいかんともしがたいものでした。「サーバー側のアプリケーションを変更すればよい」という配布の容易さ、「いろいろな環境で動作する」というメリットをもってしても、それは変わりません。

もちろん、それをただ指をくわえて見ていただけではなく、DHTMLやActiveXコントロール（ActiveXドキュメント）などの新しいテクノロジーが惜しみなく投入されました。しかし、Internet Explorer（以下IE）でしか動かなかったり、セキュリティの問題があったりしたため、それほどの追い風とはなりませんでした。また、表示を受け持つHTMLと、ビジネスロジックであるサーバーサイドスクリプトがひとつのファイルに同居し、「保守性や拡張性

に難がある」というASPのデメリットもありました。

◎リッチクライアントの登場

しかしASPのデメリットであった開発のしづらさは、.NET Frameworkとともに現われたASP.NETで見事解消されました。ASP.NETではHTMLとコントロール部分、ビジネスロジックを分離し、拡張性と保守性を見事にクリアしたのです。しかし、表現力不足という点については、ASP.NETになってもなんら変わることがありません。それは、表現力不足がHTMLの持つ制限だったためです。

そこで出てきたのが「リッチクライアント」と呼ばれるアプリケーションモデルです。クライアントとサーバーの通信にはHTTPなどを使い、アプリケーションの配布自体はオンデマンドのような形でダウンロードする、という仕組みです[注1]。

このような背景を踏まえ、今回紹介する「ClickOnce」はリッチクライアントに対する、Microsoftが出してきた答えのひとつです。クライアント／サーバー型が持つ表現力の高さと、Webアプリケーションが持つ配布のしやすさを備える、将来の主流候補のひとつ……と紹介していくつもりだったのですが、そうすんなりといかないのが実社会の面白いところです。

注1)「リッチクライアント」という用語は、いくつかの意味で使われることがあります。一般的には、このような意味合いで使われますが、Microsoftの用語としては「Windowsフォーム」を意味します。

◎Ajaxによる新しいWebアプリケーション

技術的にはさほど新しくはないものの、最近とくに注目を集めるようになったテクノロジー「Ajax（Asynchronous JavaScript and XML)」の登場が、Webアプリケーションの世界に新しい波を起こし始めました。MicrosoftもAjaxには強く注目しているようで、ASP.NET 2.0用にAtlasという専用のフレームワークを用意しています。

・Atlas
http://beta.asp.net/default.aspx?tabindex=7&tabid=47

現時点では、それぞれのアプリケーションモデルにはメリットとデメリットがあり（表1)、「ひとり勝ち」という状況には至っていません。いずれかのアプリケーションモデルが勝利するのか、それとも共存していくのか、今の時点ではまったくわかりません。どちらにせよ、現時点で結論が出ていない以上、いずれかひとつのアプリケーションモデルだけを選ぶことはできないでしょう。開発者にとっては、「学ぶことが増える」という相変わらずの状況が続いていくことになりそうです。

.NETアプリケーション開発における「配布」

.NET Framework上でアプリケーションを開発するようになって、「一番よかった点」は何でしょうか？　もちろん、統一的なクラスライブラリや、C#のような最新のプログラミング言語を使って開発できる、といったメリットがあります。それとともに、見過ごされがちではありますが、「アプリケーションの配布が楽になった」という点を忘れてはいけないでしょう。

.NETアプリケーションは、当然のことながら.NET Framework上でしか動作しません。しかし、逆の言い方をすれば、.NET Frameworkさえインストールされていれば、どんな環境でも動くということになります。

そしてさらに2つ、それまでのアプリケーション配布に比べて優位な点があります。ひとつ目は、レジストリなどのオペレーティングシステムリソースへの依存がないこと。2つ目は、データベースアクセスなどの基本コンポーネントは.NET Frameworkとともにすでにインストールされているということです。

もちろん、このほかにSide-By-Side

表1：アプリケーションモデルごとのメリット／デメリット

	開発のしやすさ	配布	表現力	動作環境
Windows Form（クライアント／サーバー）	比較的簡単	難しい	かなり高い	Windowsのみ
Windows Form リッチクライアント	比較的簡単	簡単	かなり高い	Windowsのみ
Webアプリケーション	比較的簡単	簡単	低い	ブラウザに依存
Webアプリ＆Ajax	かなり難しい	簡単	高い	ブラウザに依存