特集2

見えてきたマイクロソフトの次の一手

次期Windows OS “Longhorn”と新しいVisual Studio .NET “Whidbey”が指し示す未来

2年後にリリースされると言われている“Longhorn”は、上から下まで.NET Frameworkで構成された、これまでとはまったく思考を変えたOSだ。それだけにα版であるにもかかわらず、次期開発環境である“Whidbey”やデータベースサーバー“Yukon”もかすむほどの存在感を示している。とはいえ、開発者が知らなければならないことは、たったひとつのキーワードだ。それは.NET Frameworkだ。これさえ飲み込めれば、LonghornもWhidbeyも、実はそれほど難解なプロダクトではない。

Part 1 Longhorn～Avalon、WinFS、Indigo～

新OSは新テクノロジーがてんこ盛り

すでに雑誌をはじめWebなどで報じられているとおり、2003年10月のPDC（Professional Developers Conference）では、次期Windows OSであるコードネーム“Longhorn”のお披露目が行なわれ、同時にVisual Studio .NETの2004年バージョンである“Whidbey”、SQL Server 2000の後継バージョン“Yukon”（いずれもコードネーム）も姿を現わした。

本特集ではLonghornの2003年11月現段階での概要と、それが開発者にとって何を意味するのか、また、Longhornを睨んで開発されつつあるWhidbeyの主な新機能とトピックを中心に紹介してゆく。

Longhornとはなんだ？

まずは図1を見てほしい。これもすでにさまざまなところで紹介されているため、もはや見慣れた方もいるだろう。これはLonghornの基本的なアーキテクチャとして取り上げられる図だが、やはりこれがなければ話にならない。

Longhornは、OSが提供すべきシステムを「（広義の）ユーザーインターフェイス」「ファイルシステム」「ネットワーク」の3種類に大きく分類し、それぞれが共通に使用するシステムとは別に纏め上げている。

図1の「Fundamentals」はその共通部分にあたるところで、カーネルモードやその他の基本的な機能を提供する。

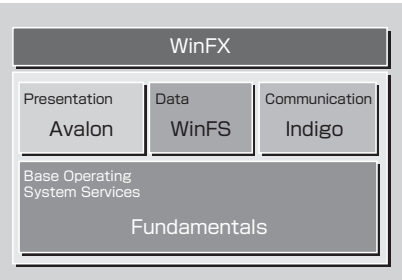
その上に、

Avalon：ユーザーインターフェイス
WinFS：ファイルシステム
Indigo：コミュニケーション

として各システムが搭載されることになる。

Fundamentalsはもとより、上位レイヤーにあたるAvalonやWinFS、Indigoなどは、（Fundamentalsの一部を除き）完全にオブジェクトとして提供されることになる。たとえば、表示関係では「GDI/GDI+」や「Direct 3D Graphics」であったり、ネットワーク関係であればTCP／UDP／IPCといった各種プロトコルのリスナーや、SIP、ネイティブなWiFi、PNRPといったネットワークサービスだ。さらに、この層にはメモリ

図1：Longhornのアーキテクチャ



マネージャーやシリアライゼーション、セキュリティ、コードエグゼクションが実装されることになる。これらはCLRであり、すなわち.NET Frameworkだ。

また、Fundamentalsの上に乗る、たとえばAvalonにはDocumentクラスやUIクラス、Mediaクラスなどが実装され、WindowsフォームやASP.NETなどのUIや、アプリケーションサービス、あるいは各種コントロールなどが提供される。

同様にWinFSではADO.NETやスキーマ、データモデルなどが、そしてIndigoではプロトコルを担当するTransport Channelやポートを担当するCommunications Managerといったコネクティビティ、キューイングやルーティングなどを担当するメッセージサービス、トランザクションを担当するシステムサービスなどが実装される。

このような形で配備された各オブジェクトが、Fundamentalsの上に乗るオブジェクトなどとおしゃべりすることによって、OSとしての機能を実現することになる。

言葉にすると非常にわかりにくいが、要は、図1の各ボックスの中には、モダンなPCを構成する上で必要十分なオブジェクトがごっそり詰まっていると思えばよい。

そしてそれらはすべて.NET Frameworkのアーキテクチャデザインを流用／拡張しているのである。つまり、OS自体がマネージドなクラスライブラリによって構成され、これまでのようなWin32 APIはWinFSのServices名前空間の下にあるSynchronaizationに下位互換という形で名前をとどめるだけとなっている。

ポイントは、

Win32 APIからマネージドクラスライブラリ（WinFX）へ

だ。

GUIの極点にむけて～Avalon

Avalonにおける開発者にとっての最大のトピックは、

Windowsアプリケーションでもコードビハインド

にある。

ASP.NETでは開発環境においてロジックとUI生成を分離し、よりロジックに注力できるような開発スタイルへの移行を促進したが、それと同じことをWindowsアプリケーションでも行なえるようにするというのがAvalonの特徴だ。

これを可能にしているのが、XAML（ザムル）ファイルであり、これはたやすく推測できるとおり、Extensible Application Markup Languageの略。このファイルでボタンやテキストボックスといったコントロールの見た目を制御することになる。記述する内容はリスト1のような、いわゆるテキスト形式のもの。

WebブラウザのMozillaなどで使用されているXULのマイクロソフト版と考えればイメージしやすいかもしれない。ただし、XULはXMLタグにMozillaのコントロールなどのウィジェットをマップするようにデザインされており、

リスト1：XAMLの例

```
<Window
   xmlns="http://schemas.microsoft.com/2003/xaml"
   Visible="true"
>
   <SimpleText Foreground="DarkRed" FontSize="24">
      Hello World!!
   </SimpleText>
</Window>
```

Mozillaによって描画されるが、XAMLはAvalonが提供するウィジェットをLonghornがマップし描画することになる。つまりWindowsで走るアプリケーションであればどんなものでもその見た目を制御できるわけだ。

ちなみにリスト1のXAMLをsimp1.xamlとして保存し、コマンドラインから、

```
C:> simp1.xaml
```

として起動した画面が図2。コンパイルすることもなく、ウィンドウを生成し、おなじみ「Hello World!!」を表示した。

これだけではあまりにもショボイので、ボタンコントロールを使ったウィンドウを2つ表示するXAMLを紹介する。リスト2がそれ。

<Button></Button>タグで囲まれた文字列がボタンのテキストとなり、タグ内のプロパティによって表示スタイルを変更することができる。スタイル自体は「This」という名で登録している。ここではバックグラウンドを赤に、文字サイズを24ポイントに設定している。

これをふたたびコマンドラインから指定して起動したのが図3。

表示された2つのボタンの下に色がつ

図2：Avalonの機能を使ってHello World!