特集2 Part 1

楽して開発！ 移行ツールのすべて

Visual BasicからVisual Basic .NETへ

膨大なVisual Basicアプリケーションをどうする？

なんとか楽してマイグレーションを果たしたい。Visual Basic .NETをイチから覚える時間がない……。
Visual Studio .NETに搭載されているアップグレードウィザードは、そんな悩みをどこまで解決してくれるのだろう？

アップグレードウィザードはアップしたか？

Conversion/Migration

デフォルトの機能を使用してみる

Visual Studio.NET 2003を起動し、スタートページから［プロジェクトを開く］ボタンを押して、VB6.0のプロジェクトファイルを開くと、Visual Studio .NET 2002と同様、アップグレードウィザードが起動し（図1）、遮二無二VBプロジェクトを.NETソリューションへと変換してくれます。流れとしては、まずアップグレードが可能かどうかをチェックし、可能であれば本格的にアップグレードを開始してもよいかどうかを聞いてきます。あとは基本的にアップグレード後のプロジェクトの種類（exeかdllまたはカスタムコントロールか）と.NET対応プロジェクトをどこに置くかを設定する以外は、おまかせです。

アップグレードが終了すると、ソリューションエクスプローラにはアップグレード後のプロジェクトが表示され（図2）、タスク一覧には手を入れなければならない箇所を表示してくれます（図3）。また、ソリューションエクスプローラに表示されている「_UpgradeReport.htm」には、タスク一覧だけでは不明な箇所に対するヘルプへのポインタが記載され、どこから手をつけるべきか悩んでいるユーザーに対して手助けを提供してくれます。

今回、アップグレードに使用したVB6.0アプリケーションのソース（.frm）は、およそ800行程度のカンタンなゲームプログラムです。一般的なメニューとラベルコントロールを30ほど配置し、フォームのLoadイベントとマウスのClickイベントを主体としたものですが、それだけでもかなりのエラーが発生します。

その大半は、生成したオブジェクトに、プログラムが必要とするプロパティがないことに起因するエラーなので（図4）、このあたりは、ヘルプを検索して、どう実装すればよいかを探すことになります。また、プログラム内でenumなどを使用していると、これらは全部コメントアウトされ、手動で変更する工程が増加します。

このことからわかるように、アップグレードウィザードは完全に.NET Framework上で動作するアプリケーションへとアップグレードしてくれるわけではありません。

Microsoftのチャットで「ウィザードをかけたけどうまくいかないんだ」という質問に対する答えの多くは「あくまでも補助的な役割のツー

図1：アップグレードウィザードが起動する

図2：ソリューションエクスプローラに表示されるファイル群

図3：タスク一覧には手動で修正しなければならない箇所が

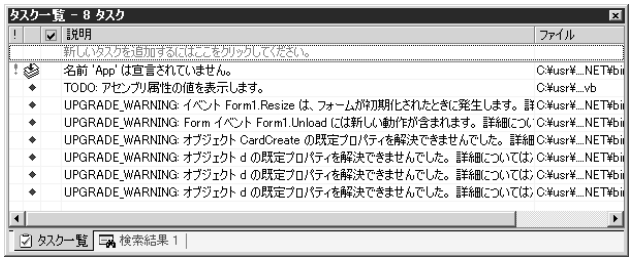
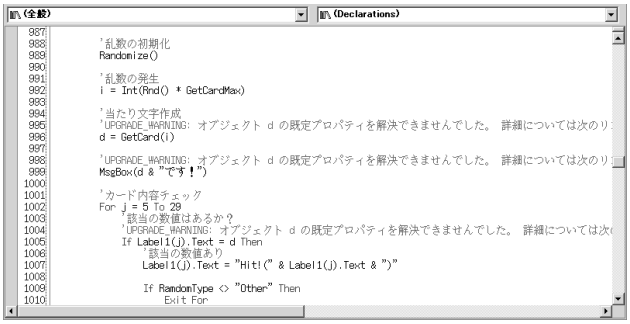


図4：手動アップグレード位置はソースにコメントとして記載される



ルだからね」という回答がなされるのもうなずけます。

ナニがサポートされないのか

アップグレードウィザードがサポートしていないテクノロジーには、いろいろあります。たとえば、OLEコンテナコントロールやVisual Basic 5.0、DDEなどは.NET Frameworkではサポートしていないので、アップグレードはできません。また、ActiveXを使用したWebテクノロジーやWebクラスは、.NET Frameworkと併用することができるので、無理やりアップグレードする必要はありません。

一方、.NET Frameworkは強烈に型を意識しているため、バリアント型はオブジェクト型へとアップグレードされますが、これによって、VB6.0のころとは微妙に異なる動作をするアプリケーションになる可能性があります。また、Win32 APIを多用していると、修正箇所が増えてゆきますし、Accessで作ったデータベースを考慮してDAOやRDOを残したプロジェクトはアップグレードしても、データバインドがADO .NETに置き換わるわけではないので、これも書き直すことになります。

アプリケーションベースで言うと、ドラッグ＆ドロップは.NET Frameworkで実装されている機能に置き換えなければなりませんし、グラフィックスメソッドもかなりの部分の書き換えが必要です。

また、ユーザーコントロールはVisual Basic .NETで使用することはできるのですが、コントロールそのものをバージョンアップすることはできません。バージョンアップするには、Visual Basic 6.0が必要になります。

いずれにしても単純にアップグレードしようとしても、なかなか前途多難です。

それでもVB6.0

現状でもVB6.0の案件を抱えているユーザーの方はまだまだ多いようです。その場合は、VB6.0でのアプリケーション構築と同時に、.NET Framworkの学習も欠かせませんが、現状でできる選択もあります。

たとえば、現時点で.NET Frameworkへの移行を視野に入れた開発を想定することもそのひとつです。それはシステムの構成そのものを見直すきっかけになるでしょう。Webブラウザをアプリケーションのフロントエンドに据えるか、むしろWindowsフォームにするのか、といったベーシックな地点からの見直しを行なうことなので、クライアントが望んでいる最適なソリューションへの近道でもあります。

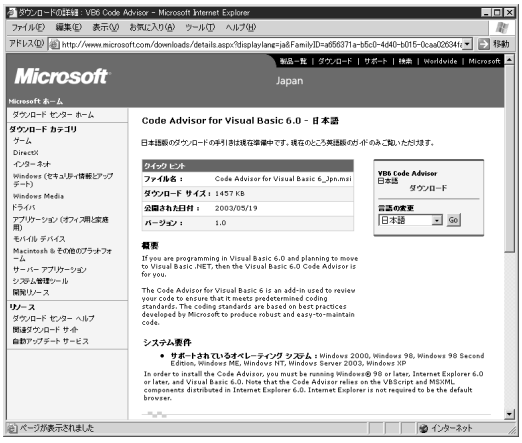
また、すでに稼動しているVB6.0のアプリケーションを、アップグレードすることを前提にサポートしているのであれば、現時点で.NET Frameworkへアップグレードしやすいソースへとブラッシュアップをかけるのもまた、ひとつの手段です。

そのための、ある程度クリティカルな部分をチェックしてくれるツールが、「Code Advisor for Visual Basic 6.0」です（図5）。これはWindows 98以降の[注1] OSで動作するVisual StudioのアドインツールでVBScriptとIE6.0に同梱されているMSXMLがあれば、動作します。このアドインをVisual Basicに追加することで、アップグレード時に問題となる点を事前に調査／報告してくれるようになります。

もちろん調査／報告だけですから、そのままでは.NET Framework上で動作するようなソースに書き換えてくれるわけではありません。あくまでもコメントを挿入し注意を喚起するだけです。また、ルール自体も、デフォルトでインストールされる標準のルール以外に新しく独自のルールを適用するためには、開発者自身が正規表現を駆使して作成しなければなりません。そのためには、VB6.0と.NET Frameworkの境界線をきちんと把握していなければならず、結果的に.NET Frameworkを学習することになるのですが。

とはいえ、これによって、自身が抱えているアプリケーションのどこに爆弾を抱えているのかが把握しやすくなりますし、それによって、学習する範囲もかなり絞ることができるはずです。

図5：Code Advisor for Visual Basic 6.0ダウンロードサイト



注1）もはやサポートされていないので、Windows 2000以降に移行すべきです。