DevTV: AutoCAD VBA から .NET 移行の基礎　（付録）

VBAの開発者の方で.NＥＴへの移行を考えているならば、COM相互運用を使用すれば思いのほか簡単に実現可能です。既にご存知のActiveX オブジェクトモデルを使用したVBAプロジェクトのVB.NETへの移行方法をご紹介いたします。

はじめに

VBA プロジェクトから VB.NET に移行する移行作業は、作業の内容をご理解いただけるとそれ程難しくはありません。この資料は DevTV プレゼンテーションで使用されるスライドとサンプルへの追加的な内容となます。是非、プレゼンテーションをご覧になった後でご使用下さい。この資料は６つの主要なトピックから構成されています:

- 基礎的なコンセプトについて（なぜ移行するのか：言語比較と IDE）
- Hello world プロジェクト（ステップ バイ　ステップ）
- ステップ バイ　ステップの移行手順
- VBA プロジェクト書き出しのヘルパーマクロの使い方
- イベントとフォームに必要な修正点
- トラブルシューティングのヒント

注意: ここでの移行に関してのご紹介は VB.NET と AutoCAD .NET API の学習を思いとどませる意図はありません。ここでは ActiveX API を使用して VBA 環境で既に作成済みのコードの再利用を最優先においております。

何故 VB.NET に移行するのか?

VBA の開発は終了しています。マイクロソフト社は “Discontinuation of the VBA Licensing Program” (http://msdn.microsoft.com/en-us/isv/bb190538.aspx) を 2007 年 7 月に発表しております。この決定には 64 ビット版の VBA が存在しないことが大きく影響しています。 （AutoCAD 64 ビット版の VBA エンジンは標準の 32 ビット版の VBA エンジンを搭載しています。AutoCAD とは別プロセスの 32 ビットコンポーネントとして実装されています）

AutoCAD 2010 より VBA Enabler をダウンロードして AutoCAD で VBA を使用可能にする必要があります。また、近い将来に VBA を AutoCAD リリースから削除することを予定しております。お持ちの VBA 資産を VB.NET に移行することを早めに開始されることを推奨いたします。この資料でご紹介するツールや移行方法が、円滑に移行作業進められるように手助けをいたします。

VBA コードの移行から得られる利点の一つは、お持ちのコードが VB.NET で動作するようになれば、WinForm、ADO.NET、AutoCAD .NET API など他の API を使用することができようになります。

補足: あなたのアプリケーションが OLE コントロールや、DDE、または DAO を使用している場合は、NSDN ライブラリの“**アップグレードを行うにあたっての注意事項**”をご覧下さい (http://msdn.microsoft.com/jp/library/ywsayxak.aspx)。

AutoCAD ActiveX API

VBA は AutoCAD ActiveX API を使用しています。マイクロソフト社の VBA を終了するとうい決定は VBA IDE が対象となっています。これは AutoCAD ActiveX API が無くなるということではありません。AutoCAD は引き続き ActiveX API を公開します。そのオブジェクト、メソッド、プロパティを VB.NET のような他の開発環境から使用できます。

これはあなたが VBA で使い慣れているオブジェクトモデルを.NET のコードでも使うことができるということです。これは COM 相互運用(COM Interop)という機能により純粋な.NET オブジェクトと同様に ActiveX オブジェクトを使用することを可能にしています。

VBA / VB.NET の比較

VBA では ActiveX API のみ使用することができます。VB.NET では ActiveX API 、AutoCAD.NET API 、そして NET フレームワークの機能を使用できます。

VBA のプロシージャを実行するには VBARUN コマンドを使用します。AutoCAD .NET のプロシージャは組み込まれたコマンドと同じ様に実行されます。 .NET のビルドされたプロジェクト (.NET アセンブリ) は NETLOAD コマンドによって AutoCAD にロードされます。この DLL はデマンドロードでもロード可能です (レジストリキー使用)。VB.NET のコードはコンパイラでコンパイルする必要があります (例： Visual Studio) 。また、.NET アセンブリを DWG ファイルに埋め込むことはサポートされていません。下記の表は VBA と VB.NET の AutoCAD で使用した場合の違いを表しています。

	VBA	VB.NET
コンパイラ	組み込み (VBAIDE)	外部ツール
ロード方法	VBALOAD または DWG に埋め込み	NETLOAD またはレジストリキー

実行方法	VBARUN	カスタムコマンド

Visual Studio: .NET IDE

マイクロソフト社は“Visual Studio 開発環境は革新的で次世代のアプリケーションを作成するソフトウェア開発者を支援する包括的なツールであり、アプリケーション開発には最適である“とコメントしています。この IDE には異なったバージョンが存在し、あなたのニーズの合った開発環境のタイプを選択できます。

Visual Studio には Express Edition という無償版が存在します。この無償版は有償版が提供する IDE にほぼ等しい機能を提供し、VB.NET コードの作成とプロジェクトをコンパイルをすることができます。ダウンロードの詳細な情報はこちらをご覧ください http://www.microsoft.com/japan/msdn/vstudio/Express/

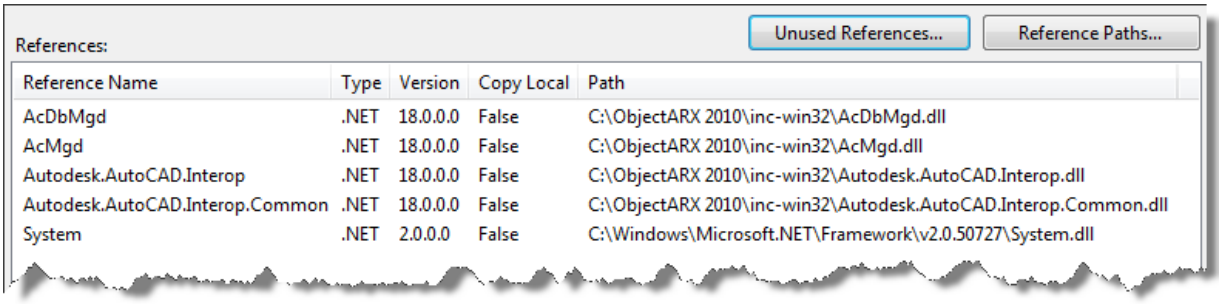
補足:.NET アプリケーションは 32 ビットと 64 ビットプラットフォーム両方で実行できます。無償版の Visual Studio は基本的には 64 ビットのコンパイルをすることはできません。Express Edition の詳細な情報にていてはこちらをご覧ください: http://msdn.microsoft.com/ja-jp/library/we1f72fb(v=VS.90).aspx.

VB.NET で ActiveX の参照追加

VB6 のように VB.NET でも *AutoCAD XXXX Type Library* と *AutoCAD/Object Common XX.X Type Library* の両方に参照の追加が必要です（注意：XXXX と XX.X はバージョン番号です）。VBAIDE で作成されたプロジェクトは自動的に AutoCAD Type Library に参照設定が行われます。それが間接的に Object Common Type Library に参照を追加します。

VB.NET プロジェクトでは必要なアセンブリを明示的に追加しなくてはなりません。Visual Studio で作成したプロジェクトでは参照追加には 3 つの選択肢があります:

1. COM オブジェクト: VBA プロジェクトで使用するのと同じものです
2. TLB ファイル: これは結果的に 1 で行うことと同じになります。しかし、手作業で*[Program Files folder]\Common Files\Autodesk Shared* フォルダに配置されている **acax18enu.tlb** と **axdb18enu.tlb** に参照を追加することができます。
3. DLL ファイル:DLL バージョンの AutoCAD COM オブジェクトを探して、どのバージョンをサポートするかを制御することができます。これらの DLL は http://www.objectarx.com よりダウンロードできる ObjectARX SDK に含まれています。この SDK を入手したら **Autodesk.AutoCAD.Interop.dll** と **Autodesk.AutoCAD.Interop.Common.dll** を見つけて下さい。ObjectARX SDK フォルダの \inc-win32 には 32 ビット版、\inc-x64 には 64 ビット版が含まれています。この 3 番目の方法がこちらで推奨する方法です。



例えば、同じマシンに AutoCAD 2008 と 2009 がインストールされているとします。この場合は、COM オブジェクト、または TLB を見つけて使用することができます。両方のバージョンをサポートする場合は 2008 の参照を使用してアプリケーションを開発することを推奨します。しかし、１．と２．の選択肢で参照を行った場合、プロジェクトは新しい方のバージョンを使用します。この例では 2009 です。明示的に 2008 の参照を行う場合は、選択肢３.で説明した方法で参照を行います。

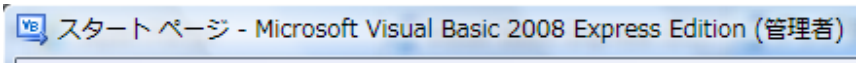
サポートをするバージョンの一番古いバージョンの参照設定でプロジェクトをコンパイルすることを推奨いたします。例えば、アプリケーションを AutoCAD 2008、 2009 、2010 で実行するとします。この場合、2008 の参照で AutoCAD 2008 と 2009 の為にコンパイルし、2010 の参照で AutoCAD 2010 のために再度コンパイルします。結果として、２つのバージョンのアプリケーションが作成されます。一つは 2008/2009 用、もう一つは 2010 用です。 **COM オブジェクト例外をキャストすることができないという** 例外が発生する場合があります。詳しくは“簡単なトラブルシューティング” 項目をご覧ください。

注意: 32 ビットと 64 ビットプラットフォームをサポートする場合は、それぞれのバージョンの参照を設定してプロジェクトをコンパイルして下さい。32 ビットバージョンの参照でコンパイルしたプロジェクトは 64 ビット版では動作しません。何故ならば AutoCAD COM オブジェクトが異なった GUID をそれぞれのプラットフォームで持っているからです。

COM Interop により VB.NET で “Hello World”

以下が ActiveX API を使用した簡単なプロジェクトを作成する手順の例です。

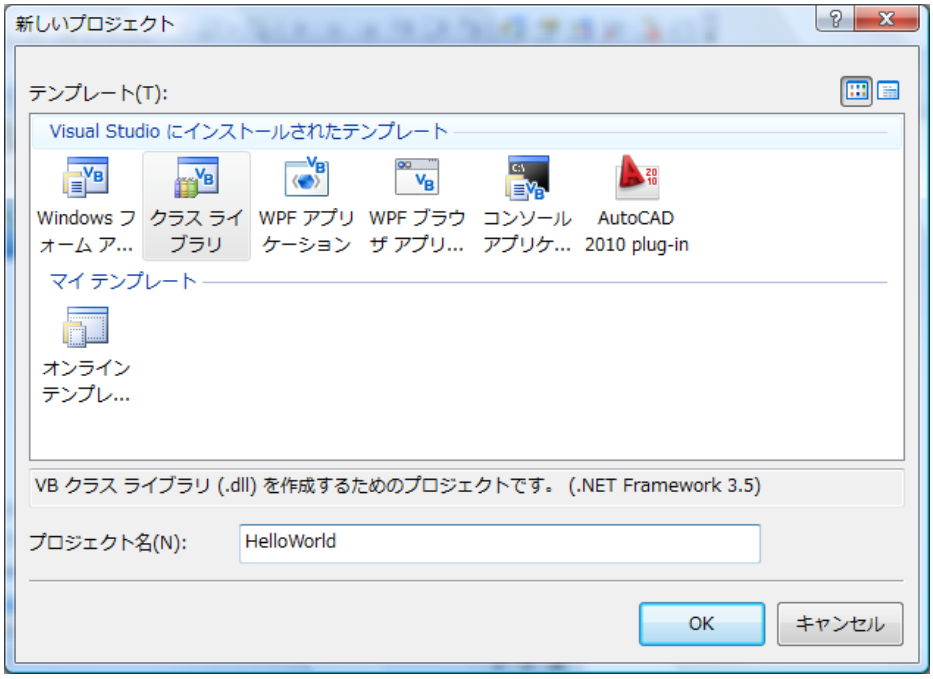
1. Visual Basic Express Edition を起動します。下記のタイトルバーでアプリケーションが起動するか確認して下さい(管理者)。起動しない場合はウィンドウズのアカウントを管理者権限にして下さい。これは必須ではなく、あくまでも推奨事項です。詳細については MSDN をご覧下さい (http://msdn.microsoft.com/jp/library/ms165100.aspx)。

2. 「ファイル」メニューで「新しいプロジェクト」を選択します。ここで Windows フォーム　アプリケーション 、またはクラス　ライブラリ を選択できます。最初のプロジェクトは AutoCAD をオートメーション可能なスタンドアローンの.exe ファイルを作成します。２つ目は.dll ファイルを作成し、AutoCAD にロードしてコマンド新たに登録することができます。

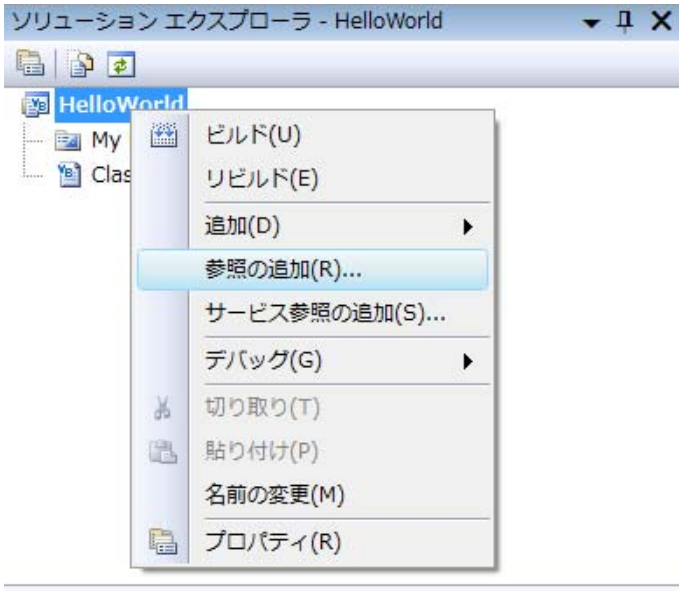
今回は VBA を移行して AutoCAD の内部で使用することが目的なので、プロジェクトの種類として下記の様にクラス　ライブラリを選びます。プロジェクト名を入力しますが、今回は “HelloWorld”として OK ボタンを押します。Visual Studio が新しいプロジェクトを作成します。

補足: 以下の “AutoCAD 2010 plu-ing” と“Autodesk Inventor Addin” は AutoCAD.NET API と Inventor COM API プロジェクトのテンプレートです。この IDE を使用してどちらも開発できます。AutoCAD .NET API プロジェクトについてはこの資料はカバーしませんが、この AutoCAD .NET Wizard は http://www.autodesk.com/developautocad　からダウンロードできます。

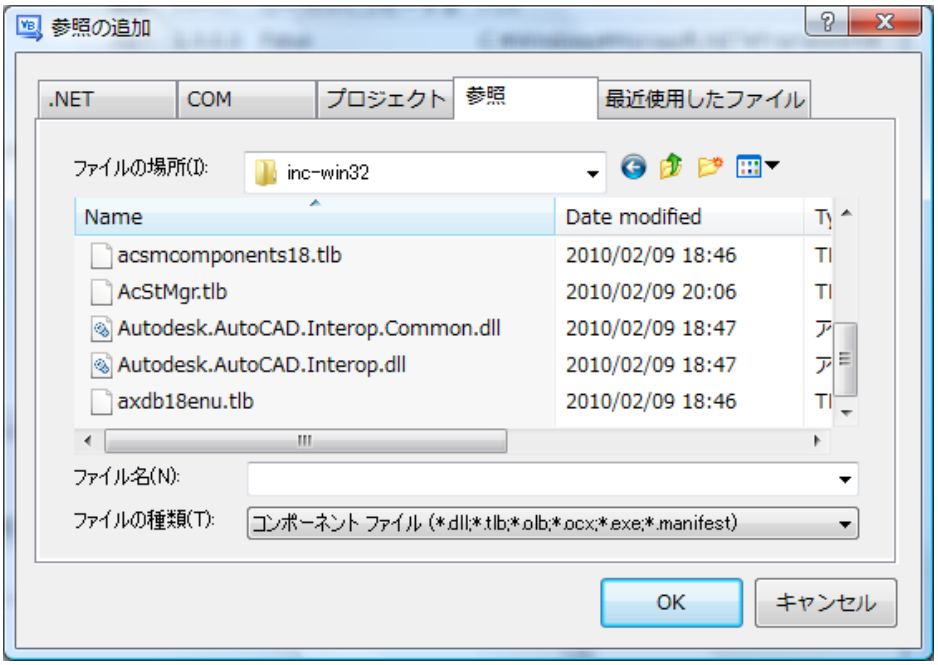


3. ソリューション　エクスポーラで **HelloWorld** プロジェクトを右クリックして「参照の追加」を選択します。プロジェクトプロパティからも追加できます。“参照”タブを選択して“追加”ボタンを押して参照を選択します。

   補足: ソリューション　エクスポーラが表示されない場合は、「表示」メニューから「ソリューション　エクスポーラ」を選択するか、“Ctrl+Alt+L ”をキー入力します。

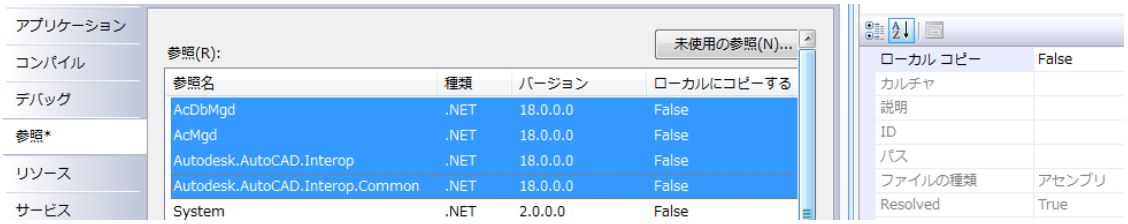


“VB.NET で ActiveX の参照追加”で推奨したように ObjectARX SDK フォルダに含まれるアセンブリを追加します。また、AutoCAD.NET API の機能を使用するので同じフォルダにある **AcMdg.dll** と **AcDbMgd.dll** の参照も追加します。



注意: アプリケーションが AutoCAD とは別プロセスで実行される場合は AcMgd と AcDbMgd に参照は追加できません。詳細については [外部アプリケーションからのオートメーション？](#) をご覧下さい。

AutoCAD に関連した参照には“ローカル　コピー”プロパティを“false”と設定して下さい。これは重複した参照とデバッグ時の問題を回避します。“ローカル　コピー”を”false”と設定するにはプロジェクト名 **HelloWorld** を右クリックして“プロパティ”で、参照を選択し、“プロパティ”ウィンドで“ローカル　コピー”プロパティを変更します。最終的には下記のようになります。



4. .NET では使用する名前空間をインポートします。これにより記述するコードの量を減らすことができます。また、移行したコードが VBA のコードと似たものとすることができます。名前空間をインポートした後の class1.vb のコードは以下の様になります:

```
'import AutoCAD Type Libray namespace
Imports Autodesk.AutoCAD.Interop
'import AutoCAD/Object Common Type Library namespace
Imports Autodesk.AutoCAD.Interop.Common

Public Class Class1

End Class
```

5. VB.NET アセンブリではプロシージャを実行する AutoCAD コマンドを定義することができます。プロシージャをコマンドとするためには以下のように属性をコマンド名と共に追加します:

```
Public Class Class1
    <Autodesk.AutoCAD.Runtime.CommandMethod("commandName")> _
    Public Sub subName()

    End Sub
End Class
```

“commandName” をユーザが AutoCAD のコマンドプロンプトから入力することで“subName” プロシージャが呼ばれます。“commandName”と “subName”は同じ名前である必要はありません。

補足: COM 名前空間でおこなったように、Imports　キーワードを使って Autodesk.AutoCAD.Runtime 名前空間をインポートすることで commandMethod 属性の記述を

“<CommandMethod(“commandName”)> _”と短くすることができます。

注意: AutoCAD のコマンドは値を返しません。Sub プロシージャの代わりに Function プロシージャを使用すると例外が発生します。

AutoCAD.NET API では VBA の ThisDrawing 変数を持ちません。AutoCAD のアクティブなドキュメントが必要なので ThisDrawing 変数を実装します。以下のコードをご覧ください。

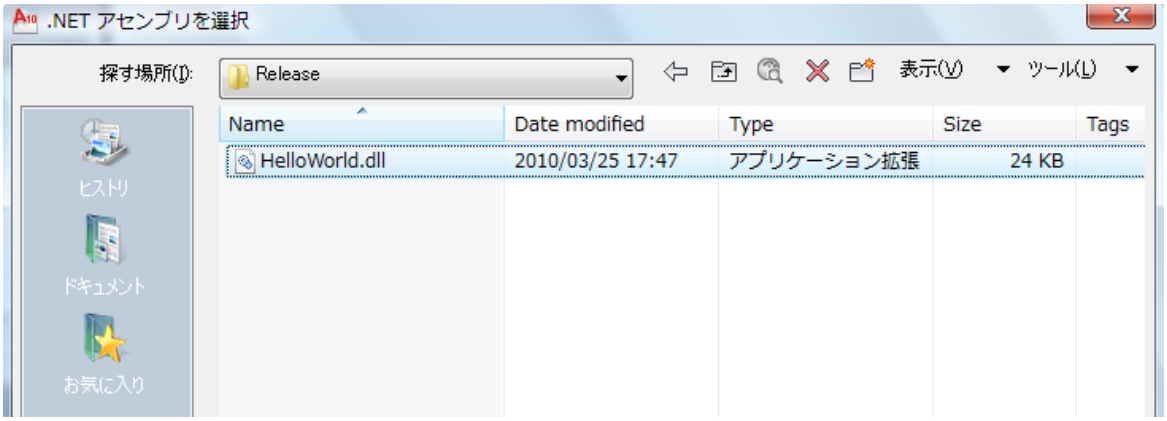
```
Public Class Class1
    <Autodesk.AutoCAD.Runtime.CommandMethod("commandName")> _
    Public Sub subName()
        'create an AcadDocument variable
        Dim ThisDrawing As AcadDocument

        'and get the Active Document
        ThisDrawing = Autodesk.AutoCAD.ApplicationServices. _
            Application.DocumentManager.MdiActiveDocument.AcadDocument

        'let's print "Hello World" on the prompt
        ThisDrawing.Utility.Prompt("Hello World from VB.NET")
    End Sub
End Class
```

6. 「ビルド」メニューから“Hello World のビルド”を選択してプロジェクトをコンパイルします。“ビルド正常終了”のメッセージが Visual Studio ステータスバーの左下に表示されます。

7. AutoCAD を起動して NETLOAD コマンドを実行します。 “.NET アセンブリを選択” ダイアログで出力フォルダの HelloWorld.dll ファイルを選択します。通常は [プロジェクトフォルダ\bin\Release フォルダに見つけることができます。 “開く”をクリックします。



プロンプトからコマンド名を入力します。AutoCAD コマンドプロンプトは以下のようになります:

コマンド: NETLOAD

コマンド: commandName
Hello World from VB.NET

ThisDrawing 変数を実装

.NET のプログラミング言語のでは get/set メソッドをプロパティに実装することができます。ThisDrawing プロパティをこの方法で作成することで既存コードの再利用をすることができます。 以下のコードは先にご紹介した **HelloWorld** サンプルを変更したものです。この変数を ReadOnly としたことにご注意下さい。このプロパティは AcadDocument クラスのインスタンスを返します。

```
Public Class Class1
   'return the Active Document
   Public ReadOnly Property ThisDrawing() As AcadDocument
      Get
         Return Autodesk.AutoCAD. _
            ApplicationServices.Application. _
            DocumentManager.MdiActiveDocument.AcadDocument
      End Get
   End Property

   <Autodesk.AutoCAD.Runtime.CommandMethod("commandName")> _
   Public Sub subName()
      'let's print "Hello World" on the prompt
      ThisDrawing.Utility.Prompt("Hello World from VB.NET")
   End Sub
End Class
```

注意: AutoCAD は MDI(Multiple Document Interface)環境をサポートしているため、ユーザが図面を切り替える都度にアクティブな図面が入れ替わります。そのため、ThisDrawing の値を変数として保存することは避けるべきです。この点については、後にイベントを使用して ThisDrawing 変数を管理する方法にて詳しくご説明します。

VB.NET から VBA コードを呼び出す

移行手順で VBA と VB.NET コードをサイドバイサイドで実行する必要がある場合があります。以下の方法で VB.NET から VBA の関数を呼び出すことができます。

```
Public Sub runMacro()
   'create an AcadApplication variable
   Dim acadApp As AcadApplication
   'get the AcadApplication
   acadApp = Autodesk.AutoCAD.ApplicationServices.Application.AcadApplication
   'and run the macro
   acadApp.RunMacro("C:\ProjectFile.dvb!ThisDrawing.MyRoutine")
End Sub
```

RunMacro メソッドは VBA ではよく知られていますが、NET に移行されています。プロジェクトが ThisDrawing プロパティ、または変数をもっていれば ThisDrawing.Application プロパティから RunMacro メソッドを呼び出すことができます。

補足: .NET プロジェクトはコマンドを定義するので、SendCommand 関数を使用してそれらのコマンドを実行することもできます。

Visual Basic Express でデバッグを有効にする

デバッグ作業なしに AutoCAD のアドインを開発することは困難です。Visual Studio Express Edition はこのオプションが UI では有効になっていません。しかし、プロジェクトファイル（.vbproj ）を直接編集することで有効にすることができます。プロジェクトファイルを編集する時は事前にバックアップをとることと、Visual Studio が実行中でないことをご確認下さい。

Notepad のようなテキストエディタでこの Hello World サンプルの **HelloWorld.vbproj** ファイルを開いて下さい。補足ですが、この変更は C#プロジェクトの.csproj ファイルにも可能です。下記の XML コードを見つけて太文字の部分 **StartAction** と **StartProgram** を追加します（希望する AutoCAD のバージョンを指定） 。

```
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
  <DebugSymbols>true</DebugSymbols>
  <DebugType>full</DebugType>
  <DefineDebug>true</DefineDebug>
  <DefineTrace>true</DefineTrace>
  <OutputPath>bin\Debug\</OutputPath>
  <DocumentationFile>HelloWorld.xml</DocumentationFile>
  <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
  <StartAction>Program</StartAction>
  <StartProgram>C:\Program Files\AutoCAD 2010\acad.exe</StartProgram>
</PropertyGroup>
```

上記の手順後に F5 キー、または「デバッグ」メニューの“デバッグ開始”を選択します。デバッグ機能は参照設定でローカルコピーの値が true となっていると使用でません。詳細は“簡単な Hello World”項目の手順３をご覧下さい。

外部アプリケーションからのオートメーションは？

アプリケーションが AutoCAD とは別の外部プロセスからオートメーションを行う場合は AcMgd と AcDbMgd アセンブリを参照することはできません。この種のアプリケーションはコマンドを定義することはできないので、これらのアセンブリの参照は必要ではありません。

補足: AutoCAD をオートメーションする場合に内部プロセスの.NET の機能を使用するためには、.NET で新たに COM インターフェースを作成してアクセスする方法があります。それについてブログにポストされています。 (http://through-the-interface.typepad.com/through_the_interface/2009/05/interfacing-an-external-com-application-with-a-net-module-in-process-to-autocad.html).

以下は移行手順の説明になりますが、あなたの外部アプリケーションが AutoCAD をオートメーションする VB6 アプリケーションであれば、VBA から VB6 に書き出す手順を省略することができます。

ステップ バイステップの移行手順

以下の項目では VBA を VB.NET に移行するための基本的な移行手順をご説明します。必要に応じて追加的な移行作業が発生することがあることをご了承下さい。

この手順をご理解いただければ、VBA から VB6 への変換ツールやイベント処理の移行などのより複雑な作業も理解しやすい筈です。作業を始める前に全ての対象ファイルのバックアップコピーをとることを忘れないで下さい。

手順 1：コードの移行準備

この後でも説明いたしますが、移行作業の主要な部分は Visual Basic アップグレードウィザードによって行われます。しかし、最終的な結果をより良いものにするために推奨いたしますプログラムの書き方があります。

遅延バインディング（Late Binding）ではなく事前バインディング（Early Binding）を使用します

これば推奨する書き方で一番重要な事です。**遅延バインディング、初期バインディングについてですが、コードで** Object や Variant で変数を定義することが**遅延バインディングとなります。プログラムは実行時に呼び出すメソッドを見つけます。**詳細については下記の資料をご覧下さい。

以前バインディングと遅延バインディング
http://msdn.microsoft.com/ja-jp/library/0tcf61s1(VS.80).aspx

例えば、以下のコードのように**遅延バインディングで変数を定義する場合は、（このコードはラベルのタイトルを変更します）コード作成時には** myLabel 変数の型情報はありません。実行時にその型が決まります。

```
Dim myLabel As Object
Set myLabel = Me.Label1
myLabel.Caption = "SomeText"
```

ウィザードが myLabel 変数の型が分からないため、適切に移行することはできません。**事前バインディング**を使用すればウィザードは適切にプロパティを移行することが出来ます。

```
Dim myLabel As Label
Set myLabel = Me.Label1
myLabel.Caption = "SomeText"
```

同じ方法が Variant 宣言や CInt、CStr などの CXxxx 関数の使用に当てはまります。以下のリンクに詳細な説明がありますので、移行作業を始める前にご一読されることを推奨いたします。

参考資料

Visual Basic .NET へのアップグレードを円滑に行うための Visual Basic 6.0 アプリケーションの準備
http://msdn.microsoft.com/jp/library/aa260644.aspx

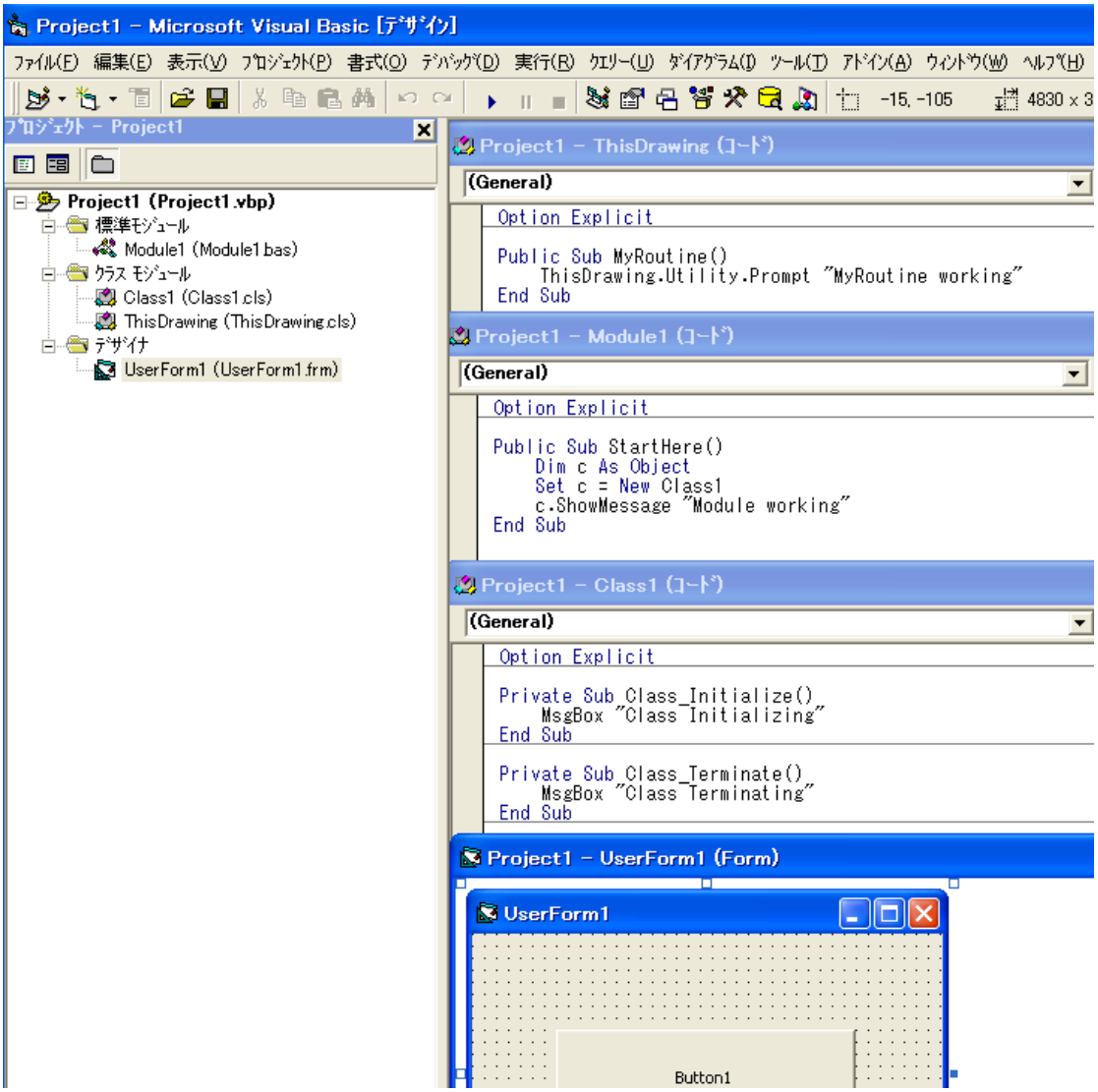
Visual Basic 6.0 アプリケーションのアップグレードの準備
http://msdn.microsoft.com/jp/library/14w905kc.aspx

Visual Studio 2005 Tools for Office への移行における VBA から Visual Basic .NET へのコード変換
http://msdn.microsoft.com/jps/library/aa537180(office.11).aspx

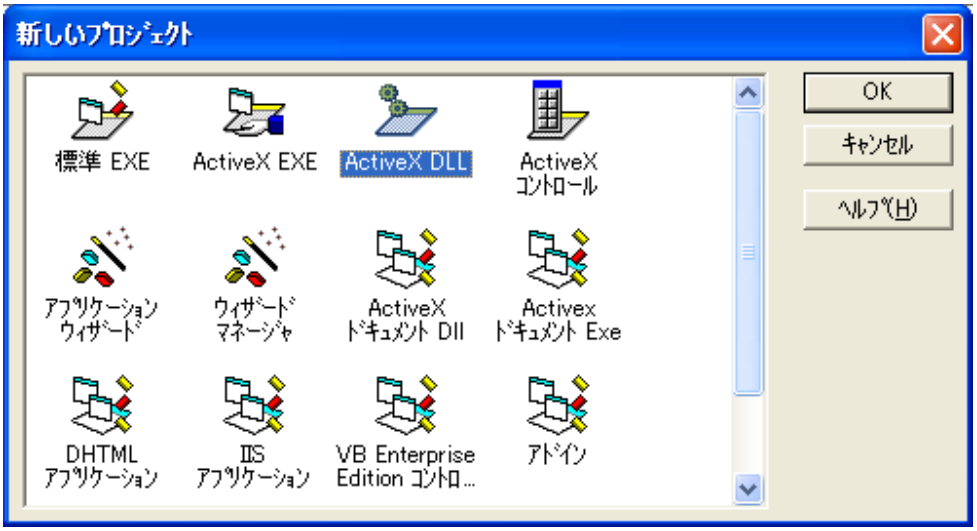
手順２: VBA プロジェクトを VB6 に書き出す

Visual Basic アップグレードウィザードは VB6 プロジェクトを VB.NET に移行することができますが、VBA プロジェクトからの移行はできません。Visual Basic アップグレードウィザードを使用するには まず VBA プロジェクトを VB6 に移行する必要があります。その作業は個々の VBA の要素を書き出して VB6 プロジェクトに読み込むという手作業になります。この手順は手作業という欠点を抱えています。そして VB6 をインストールすることも必要になります。しかし、VB6 の使用を避けることができるユーティリティツールが存在します。それについては後にご紹介いたします。

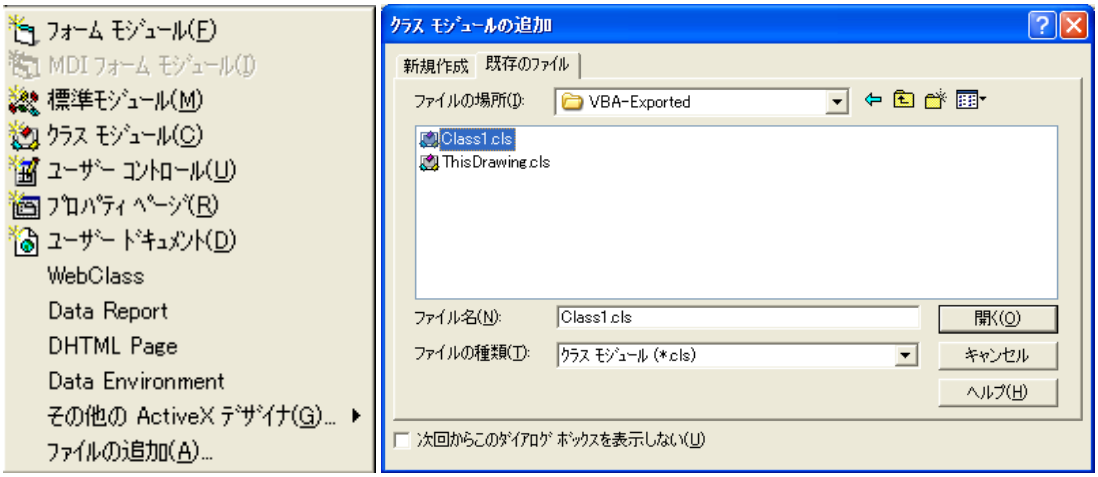
１つのモジュール、１つのクラス、１つの UserForm を含み、ThisDrawing を使用する簡単な VBA プロジェクトの例では、この個々の要素を「ファイル」－＞「ファイルの書き出し」メニューで選択して書き出すことができます。その結果は.bas と.cls ファイルになります。



Visual Basic 6.0 を起動して「ファイル」－＞「新しいプロジェクト」を選択します。次に下記のように **ActiveX DLL** プロジェクトを選択して OK をクリックします。 新しいプロジェクトが作成されます。このではこのプロジェクトのプロパティは変更しませんが、後に VB.NET にて変更します。

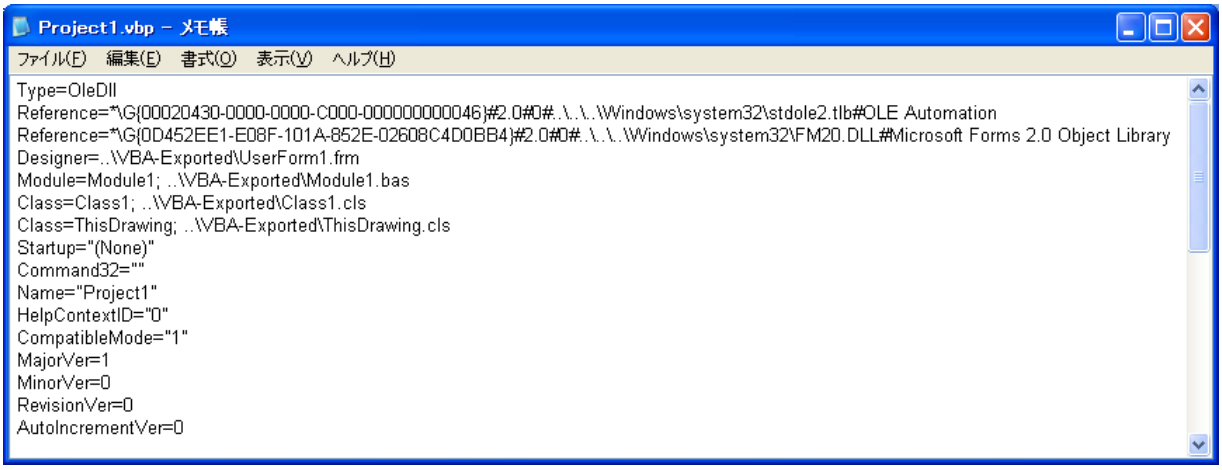


VB6 で新しく作成したプロジェクトが開かれている状態で、「プロジェクト」メニューで、VBA から書き出した各クラスに「クラスモジュール」、各モジュールに「標準モジュール」、各ユーザフォームに「フォームモジュール」を選択します。ダイアログの“既存のファイル”タブでファイルを選択します。



最後に、VB6 プロジェクトを.vpb ファイルとして保存します。このファイルを Notepad で開くとテキストファイルであることがわかります。そして.bas と.cls ファイルへの参照を含んでいることが見つかります。ご存知のようにテキストファイルは簡単に編集可能です。この後ご紹介する VBA から VB6 の変換ツールはこの編集機能を活用しています。

プロジェクトの移行準備ができました。先に進みましょう。



手順３: Visual Basic アップグレードウィザードを使う

アップグレードウィザードを使用します。ウィザードでは　“DLL/カスタム　コントロールライブラリ”プロジェクトタイプなど既定値をそのまま使用します。必要であれば保存先などを変更して下さい。

アップグレードウィザードは各 VB6 の要素に対してのファイルと、さらに３つファイルを作成します。:

- **AssemblyInfo:** このファイルはバージョン番号、詳細情報、ロードされたアセンブリなどアプリケーションの情報を取得するためのプロパティを提供します。(http://msdn.microsoft.com/library/microsoft.visualbasic.applicationservices.assemblyinfo.aspx).
- **app.config:** アプリケーション固有設定を含む基本設定ファイルです。(http://msdn.microsoft.com/jp/library/ms229689.aspx).
- **_UpgradeReport**: 移行作業についての情報を含んでいます。このリポートに目を通すことをお勧めいたします。

まず、Module1 のコードを見てみましょう。結果を実演する目的で意図的に遅延バインディングのオブジェクト“c” を残しています。VB.NET はどの型かわからないために Object タイプを作成します (この場合は**_UpgradeReport** に警告が表示されます)。アップグレードウィザードはオブジェクト“c”が ShowMessage メソッドをもっていることを知りません。

その他のコードは問題なく移行され、幾つかの文法が変わった事を除くとほとんど元のコードと変わりありません。 例えば Set キーワードの削除と丸括弧(…) が各メソッドの呼び出しに追加されています。

```
Module Module1
      Public Sub StartHere()
            Dim c As Object
            c = New Class1
            'UPGRADE_WARNING: Couldn't resolve default property of object
            'c.ShowMessage.
            c.ShowMessage("Module working")
      End Sub
End Module
```

アップグレードウィザードは幾つかの変更を Initialize と Terminate メソッドに行っています。アップグレードウィザードはこれらの古いメソッドに_Renamed 接尾辞を付けて New と Finalize メソッドから呼んでいます。

上級トピック:.NET はガーベッジコレクション機能を使用します。しかし、何時 Finalize が実行されるのかはっきり決まっていません。リソースは決まったタイミングで解放される保障はありませんのご注意下さい。

全ての.NET オブジェクトは暗黙的に、また明示的に **System.Object** クラスから派生しています。よって **Class1** は MyBase.New と MyBase.Finalize を呼ぶべきです。MSDN の資料によると Object クラスは.NET フレームワークの派生階層に存在するの全てのクラスをサポートし、派生クラスに下階層のサービスを提供します。 Object クラスは全ての.NET フレームワークに存在するクラスの基本クラスです。(http://msdn.microsoft.com/jp/library/system.object.aspx)。

補足: 自動生成されたコードに UPGRADE_NOTE コメントが追加されている部分があります。これはアップグレードウィザードが何をしたのかを明確にするためです。これらのコメントは詳細情報として MSDN Web サイトのリンクも含んでいる場合もありますが、それらはここでは削除しています。

```
Friend Class Class1
    'UPGRADE_NOTE: Class_Initialize was upgraded to Class_Initialize_Renamed.
    Private Sub Class_Initialize_Renamed()
        MsgBox("Class Initializing")
    End Sub
    Public Sub New()
        MyBase.New()
        Class_Initialize_Renamed()
    End Sub
    'UPGRADE_NOTE: Class_Terminate was upgraded to Class_Terminate_Renamed.
    Private Sub Class_Terminate_Renamed()
        MsgBox("Class Terminating")
    End Sub
    Protected Overrides Sub Finalize()
        Class_Terminate_Renamed()
        MyBase.Finalize()
    End Sub

    'UPGRADE_NOTE: str was upgraded to str_Renamed.
    Public Sub ShowMessage(ByRef str_Renamed As String)
        MsgBox(str_Renamed)
    End Sub
End Class
```

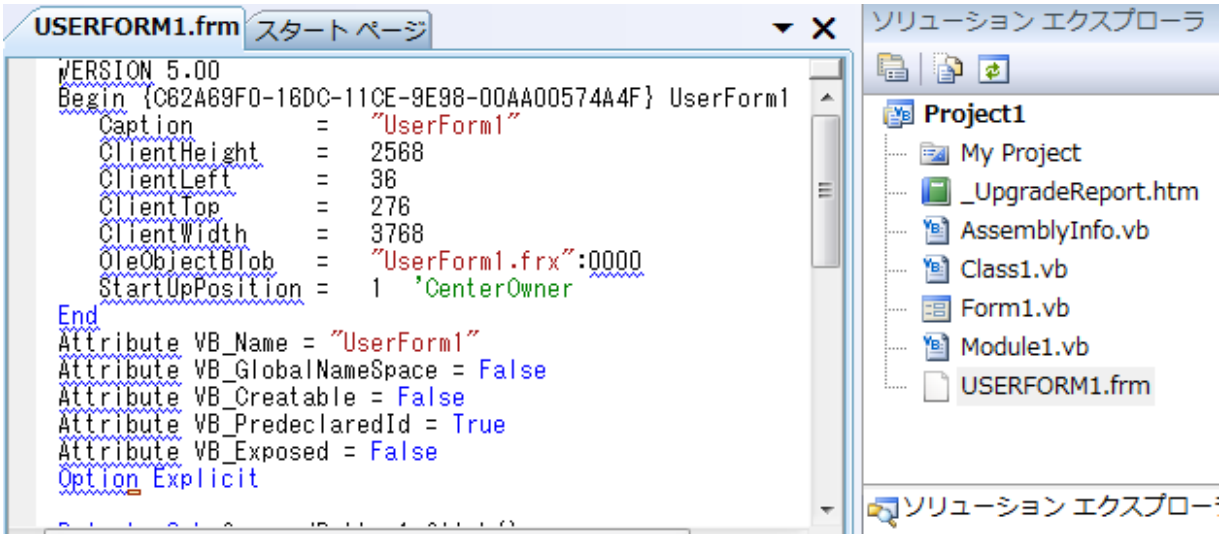
VBA の ThisDrawing オブジェクトは VBA でのみ存在します。従ってアップグレードウィザードは認識することができません。その部分には UPGRADE_WARNING コメントを追加しています。この警告は _UpgradeReport ファイルにも示されています。

アップグレードウィザードはこの問題を解決する為に以下のクロスアウトされている行を作成しています。この行は不必要なので削除しますが、それによってコンパイルエラーが発生します。その対処が必要となります。

```
<System.Runtime.InteropServices.ProgId("ThisDrawing_NET.ThisDrawing")> Public
Class ThisDrawing
    'UPGRADE_NOTE: ThisDrawing was upgraded to ThisDrawing_Renamed.
    Public Sub MyRoutine()
        Dim ThisDrawing_Renamed As Object
        'UPGRADE_WARNING: Couldn't resolve default property of object
        'ThisDrawing.Utility.
        ThisDrawing.Utility.Prompt("MyRoutine working")
    End Sub
End Class
```

注意: UPGRADE_NOTE は有益な情報をもたらしますが、UPGRADE_WARNING は通常コンパイルエラーを引き起こす問題の情報を含んでいますので、注意して確認して下さい。アップグレードリポートでそれらの場所を確認できます。

VB 6 はユーザフォームを処理することはできません。従って、アップグレードウィザードはそれらを移行しません。以下のように作成された VB.NET プロジェクトはユーザフォームの VB6 のコードを含みます。これは VBA プロジェクトから VB.NET の移行の大きな問題点です。この後ご紹介する VBA から VB6 に変換するツールでこの問題に対応しています。



手順４: 結果の修正

必要な修正は“Hello World” 項目でもご紹介いたしました。まず、２つの Interop DLL **AutoCAD.Autodesk.Interop** と **AutoCAD.Autodesk.Interop.Common** に参照を追加しました。 次に、AutoCAD.NET を使用するために **AcMgd** と **AcDbMdg** に参照追加をしました。

補足: AutoCAD.NET アセンブリ追加時に、Visual Basic が“最新バージョンの.NET フレームワークが必要です”警告する場合があります。その時は トラブルシューティング項目の“AutoCAD.NET アセンブリ (AcDbMgd または AcMdg) は最新バージョンの.NET フレームワークが必要です” トピックを参照下さい。

２つの Imports 行と ThisDrawing プロパティを追加します。そして MyRoutine メソッドをカスタムコマンドとするために CommandMethod 属性を追加します。結果は下記のようになります(アップグレードウィザードコメントは削除しています)。

```
'Import these namespaces
Imports Autodesk.AutoCAD.Interop
Imports Autodesk.AutoCAD.Interop.Common

'Add the ThisDrawing property
Public Class ThisDrawing
    Public ReadOnly Property ThisDrawing() As AcadDocument
        Get
            Return Autodesk.AutoCAD. _
                ApplicationServices.Application. _
                DocumentManager.MdiActiveDocument.AcadDocument
        End Get
    End Property
    'Add this attribute to mark this sub routine as a command
    <Autodesk.AutoCAD.Runtime.CommandMethod("myRoutine")> _
    Public Sub MyRoutine()
        ThisDrawing.Utility.Prompt("MyRoutine working")
    End Sub
End Class
```

補足: 場合によっては ThisDrawing 変数は他のモジュールから使用されることがあります。その場合は“'Application' は'ThisDrawing'のメンバではありません”というコンパイルエラーとなります。解決策は幾つかありますが、下記のように適切な名前のプロパティを作成してそのタイプを返すことです。

```
Imports Autodesk.AutoCAD.Interop

Public Class ThisDrawing
    '...
    'the rest of your ThisDrawing class
    '...
    Public Shared ReadOnly Property _
    Application() As AcadApplication
        Get
            Return Autodesk.AutoCAD. _
            ApplicationServices. _
            Application.AcadApplication
        End Get
    End Property
End Class

'...
'somewhere else in your project
Public Module Module1
    Public Sub someMethod()
        'call the ThisDrawing.Application
        ThisDrawing.Application.Visible = True
    End Sub
End Module
```

また、Module は Public として更新されません。従ってコマンドは AutoCAD に認識されません。コマンドを実行するためには Public として定義して CommandMethod 属性を追加してコマンドの定義を行います。

```
Public Module Module1
    <Autodesk.AutoCAD.Runtime.CommandMethod("StartHere")> _
    Public Sub StartHere()
        <<The rest of the code have not changed>>
```

VBA から VB6 への変換: “マジック” ヘルパーツール

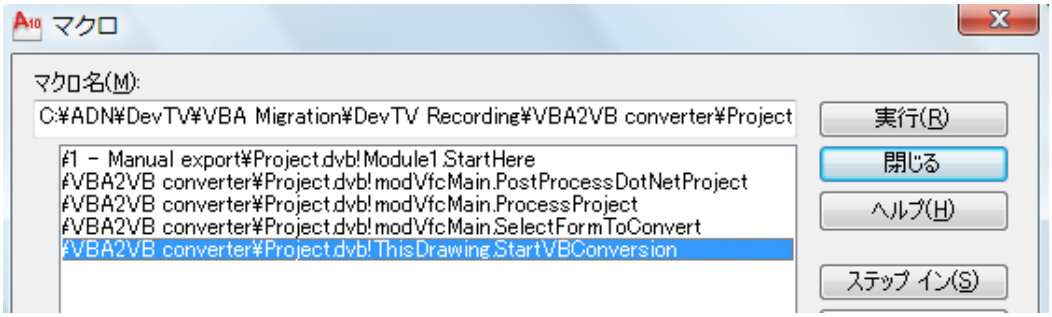
VBA から VB6 の変換ツールは VBA プロジェクトを VB6 プロジェクト形式に書き出し、Visual Basic アップグレードウィザードで使用可能にするためのマクロです。結果として VB6 をインストールする必要がありません。これらのプロジェクトファイルがテキスト形式であり、外部から編集可能であることにあることに着目しました。　個々の VBA ファイルを書き出し、新たに作成した VB6 プロジェクトに自動で読み込みます。その為、手順２の手作業を省略することができます。この変換ツールのオリジナルバージョンは Leslie Lowe 氏(mookiha@hotmail.com) によって作成されました。以下は追加された新しい機能です。オリジナル版は http://discussion.autodesk.com/forums/thread.jspa?threadID=410953　より入手できます。

このツールを使用することの一番の利点はユーザフォームを VB6 フォームに変換してくれることです。これによりアップグレードウィザードが WinForm と呼ばれる VB.NET フォームとコントロールに移行することを可能にします。このツールは一般的によく使われる限られた種類のコントロールをサポートします。以下がそのコントロールです:

- MSForms.Label
- MSForms.TextBox
- MSForms.CheckBox
- MSForms.OptionButton
- MSForms.CommandButton
- MSForms.ToggleButton
- MSForms.Image
- MSForms.ListBox
- MSForms.ComboBox
- MSForms.ScrollBar
- MSForms.Frame

注意: このツールは MSForm コントロール、TabStrip、Multipage、 SpinButton、またはその他の外部コントロールを処理できません。 それらは手作業で移行する必要があります。

このツールを使用するにあたっては、移行したいプロジェクトと **VBA_to_VB6_Converter.dvd** を VBALOAD でロードします。そして 下記のように **StartVBConversion** マクロを VBARUN で実行するとダイアログボックスが表示されます。



ツールは下記のように３つセクションに分かれています。それぞれが移行手順に該当します。

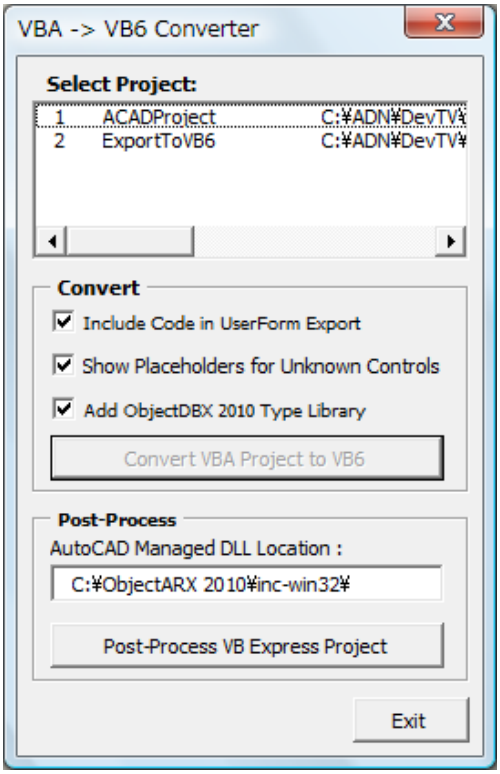
- 最初に変換するプロジェクトを選択します。それは VBALOAD ですでにロードされている必要があります。このリストは ExportToVB6 プロジェクトを表示していますが、これが変換マクロです。

- 次にいくつかの設定を変更します:
  - UserForm を移行に含めるかどうか、例えば、サポートされていないコントロールを数多く含んでいる場合は、この設定は使用するべきかどうかなどの判断が必要です。
  - マクロが変換できないコントロールの配置場所を確保する
  - 必要であれば ObjectDBX タイプライブラリの参照を追加

  プロジェクトに応じた基本設定後に “Convert VBA project to VB6” ボタンを押します。作成されたプロジェクトは [VBA プロジェクトフォルダ]\VB6Conv フォルダに配置されます。

- アップグレードウィザード使用後に 3 番目のセクションは VB.NET プロジェクト設定を行います。その機能を使用するには、まず VB.NET プロジェクトを閉じます。そして “ Post-Process VB Express Project “ ボタンを押し、アップデート済みの.vbproj ファイルを選択します。それにより下記の変更が行われます:
  - 対象のフレーワークを.NET Framework を 3.5 に設定することで “トラブルシューティング”で説明している警告メッセージを抑制することができます。
  - デバッグと F5 ショートカットキーを Express Edition で有効にします。
  - ターゲットプラットフォームを “Any CPU” に設定して Express Edition の 32 ビット制限を取り払います。
  - ObjectARX SDK フォルダに含まれる必要なアセンブルの参照を追加します。SDK の場所が正しく設定されているか確認して下さい。

  この参照設定は後で変更可能です。COM Interop バージョンの変更については “VB.NET で ActiveX の参照追加” をご覧ください。



イベントについて

VB.NET でのイベント処理は VB6 と VBA によく似ています。変数が **WithEvents** キーワードで宣言されている場合はメソッドがイベントを処理できます。しかし、イベントを取得するメソッドは **Handles** キーワードで明示されように修正する必要があります。一般的にアップグレードウィザードは文法が似ているのでイベントを処理する変数には対応します。しかし ThisDrawing のイベントは移行できません。

VBA では ThisDrawing 変数は特別な特徴をもっています。それはカレント図面に対応してコンテキストを自動的に変えます。基本的に、実行環境が変数を更新して新しいドキュメントのイベントの設定を行います。例えば、ThisDrawing モジュールの以下の VBA コードをご覧下さい。全てのドキュメントに問題なく実行することができます。

```
Private Sub AcadDocument_BeginCommand(ByVal CommandName As String)
   ThisDrawing.Utility.Prompt "(Begin command " & CommandName & “)”
End Sub
Private Sub AcadDocument_EndCommand(ByVal CommandName As String)
   ThisDrawing.Utility.Prompt "(End command " & CommandName & “)”
End Sub
```

VBA から VB6 への変換ツールを使用してアップグレードウィザードで更新後、ThisDrawing に対応した変更を行うと上記の VBA は以下の VB.NET になります。イベントを適切に実装するためには単なる

ThisDrawing プロパティ以上のものが必要です。そのため以前の例で使用したようなプロパティは実装することができません。

```
Public Class ThisDrawing
    'UPGRADE_NOTE: ThisDrawing was upgraded to ThisDrawing_Renamed.
    Private Sub AcadDocument_BeginCommand(ByVal CommandName As String)
        'UPGRADE_WARNING: Couldn't resolve default property of object
        'ThisDrawing.Utility.
        ThisDrawing.Utility.Prompt("(Begin command " & CommandName & ")")
    End Sub

    'UPGRADE_NOTE: ThisDrawing was upgraded to ThisDrawing_Renamed.
    Private Sub AcadDocument_EndCommand(ByVal CommandName As String)
        'UPGRADE_WARNING: Couldn't resolve default property of object
        'ThisDrawing.Utility.
        ThisDrawing.Utility.Prompt("(End command " & CommandName & ")")
    End Sub
End Class
```

上記のコードを直すには幾つかの手順が必要です。

1. COM Interop に必要な名前空間をインポートします。既にご説明しているように、これによって VB.NET のコードをより VBA に似たのにすることができ、結果として手作業による変更を減らすことができます。また、この場合は ApplicationServices 名前空間をインポートすることでタイプするコードの量を減らすことができます。

2. ThisDrawing 変数はアクティブなドキュメントを示すように、AcadDocument クラスの変数を作成してその値を.NET のアプリケーションオブジェクトを使用して設定します。前回は変数として作成するしないで読み込み専用のプロパティとしましたが、VBA のイベントコードのように **WithEvents** キーワード（プロパティには使用不可）を使用して宣言することが必要です。この理由で変数を使用します。

3. アクティブドキュメントの変数はユーザがドキュメントを切り替える都度に変更されなければなりません。この変化を監視するために.NET のアプリケーションオブジェクトから取得する DocumentCollection オブジェクトを使用してドキュメントのコレクションにアクセスできます。

4. DocumentCollection オブジェクトを使用して DocumentActivated イベントのイベントハンドラを作成します。このイベントにて ThisDrawing 変数の値を更新します。

5. アップデートウィザードは ThisDrawing を認識しません。Handles キーワード は手作業で追加する必要があります。

```
'<< Step 1 >>
'Import these namespaces
Imports Autodesk.AutoCAD.Interop
Imports Autodesk.AutoCAD.Interop.Common
'Another namespace – recommended here
Imports Autodesk.AutoCAD.ApplicationServices

Public Class ThisDrawing
```

```
    '<< Step 2 >>
    'Declare a ThisDrawing variable WithEvents
    Private WithEvents ThisDrawing As AcadDocument = _
        Application.DocumentManager.MdiActiveDocument.AcadDocument

    '<< Step 3 >>
    'We also need to keep track of the active document, so let's
    'get the DocumentManager, which control all opened documents
    Private WithEvents Docs As DocumentCollection = _
       Application.DocumentManager

    '<< Step 4 >>
    'every time the active document change, store it at thisDrawing
    Private Sub Docs_DocumentActivated(ByVal sender As Object, _
                                       ByVal e As DocumentCollectionEventArgs) _
                                       Handles Docs.DocumentActivated
        ThisDrawing = e.Document.AcadDocument
    End Sub

    '<< Step 5 >>
    'Append the Handles keyword for the migrated code
    Private Sub AcadDocument_BeginCommand(ByVal CommandName As String) _
                                          Handles ThisDrawing.BeginCommand
        ThisDrawing.Utility.Prompt("(Begin command " & CommandName & ")")
    End Sub
    Private Sub AcadDocument_EndCommand(ByVal CommandName As String) _
                                        Handles ThisDrawing.EndCommand
        ThisDrawing.Utility.Prompt("(End command " & CommandName & ")")
    End Sub
End Class
```

上級トピック: 上記のコードはある必要条件があります。VBA と同様に、ロード後にコマンド開始と終了のイベントを監視しなくてはなりません。しかし、その振る舞いは起こらないかもしれません。なぜならば ThisDrawing 変数がまず初期化されなければならないからです。この場合はクラスがインスタンス化されるまでそれは行われません。クラスにコマンドに含まれる場合は、コマンドが最初に実行された時に初めて初期化されます。NET アセンブリでは IExtensionApplication インターフェースがアセンブリロード時に実行されます。よって上記のクラス、また新たにクラスを作成してそこで必要なオブジェクトを初期化することができます。以下のコードはこのインターフェースのシグニチャを示したものです。 Initialize と Terminate メソッドを宣言する必要があります。

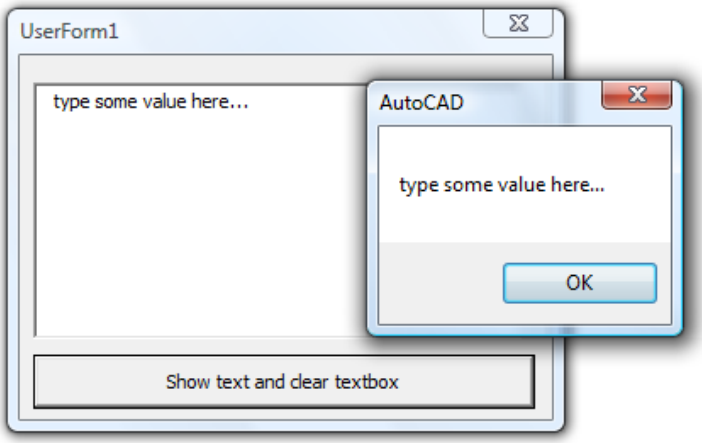
```
Public Class ThisDrawing
    Implements Autodesk.AutoCAD.Runtime.IExtensionApplication

    'Required when implementing the IExtensionApplication interface
    Public Sub Initialize() Implements _
        Autodesk.AutoCAD.Runtime.IExtensionApplication.Initialize
    End Sub
    Public Sub Terminate() Implements _
        Autodesk.AutoCAD.Runtime.IExtensionApplication.Terminate
    End Sub
```

```
    '...
    'the rest of the code . . .
    '...
```

ユーザフォームの移行

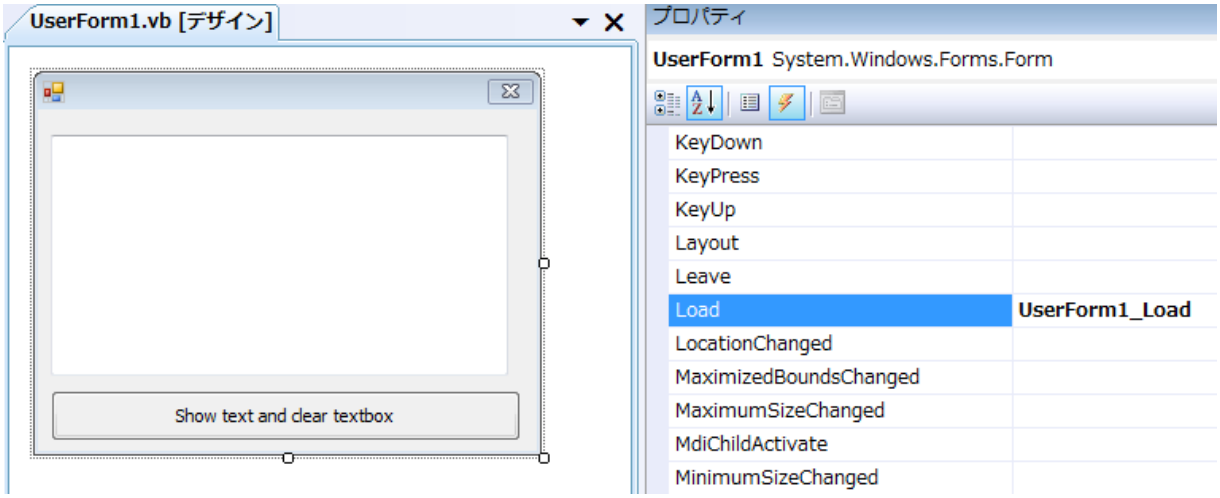
ユーザフォームのプロパティに関しては VBA から VB6 変換ツールは問題なく動作します。
しかし、追加的な情報が必要です。以下の例は、文字ボックスとボタンを持つシンプルなユーザフォームでフォームが初期化される時に文字ボックスに初期値が設定され、ユーザがボタンをクリックすると文字ボックスの文字が表示されてから消去されというものです。




```
Private Sub CommandButton1_Click()
   Call showTextboxText(TextBox1)
   TextBox1.Text = ""
End Sub
Private Sub showTextboxText(tbox As TextBox)
   MsgBox tbox.Text
End Sub
Private Sub UserForm_Initialize()
   TextBox1.Text = "The initial value"
End Sub
```

VBA から VB6 への変換ツール実行、Visual Basic アップグレードウィザード実行すると.NET WinForm は元のものと大変類似しています。しかし、フォームのイベントに変更が必要です。

Visual Basic はフォームとコントロールのイベントに独特な処理をおこないます。使用するには、フォームまたはコントロールを選択し、プロパティウィンドウのイベントアイコン ( ) をクリックして使用可能なイベントを全てリストします。 **Load** をマウスでダブルクリックするとイベントを処理するメソッドが新たに作成されます。



コードでは古い初期化メソッドを呼ぶか、元のコードを新しいメソッドにコピーして貼り付けすることができます。結果は下記のとおりです。**Handles** キーワードがボタンクリックイベントに追加されています。今回はイベントが基本クラスで宣言されているので **MyBase** キーワードが使用されています。

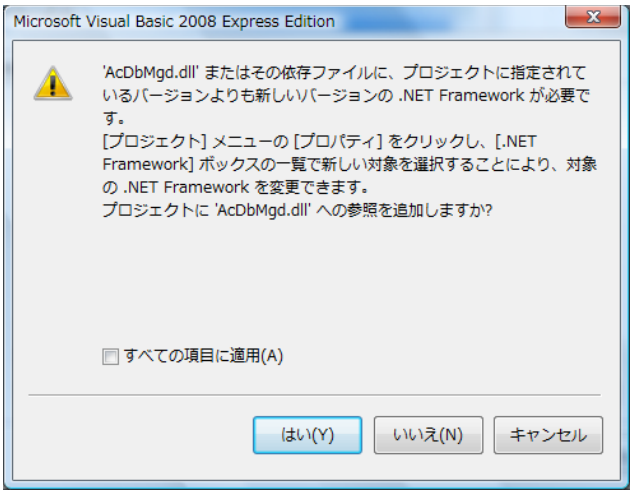
```
Friend Class UserForm1
    Inherits System.Windows.Forms.Form
    Private Sub CommandButton1_Click(ByVal eventSender As System.Object, _
                                     ByVal eventArgs As System.EventArgs) _
                                     Handles CommandButton1.Click
        Call showTextboxText(TextBox1)
        TextBox1.Text = ""
    End Sub
    Private Sub showTextboxText(ByRef tbox As System.Windows.Forms.TextBox)
        MsgBox(tbox.Text)
    End Sub
    'This initialize will not work, we need to create this event again
    Private Sub UserForm_Initialize()
        TextBox1.Text = "The initial value"
    End Sub
    'Here is the new Load event that replaces the Initialize
    Private Sub UserForm1_Load(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) _
                               Handles MyBase.Load
        'call the old initialize (or copy/paste the code here)
        UserForm_Initialize()
    End Sub
End Class
```

補足: VB.NET プロジェクトがイベントを適切に処理するように、全て **Handles** キーワードで実装されているか確認して下さい。

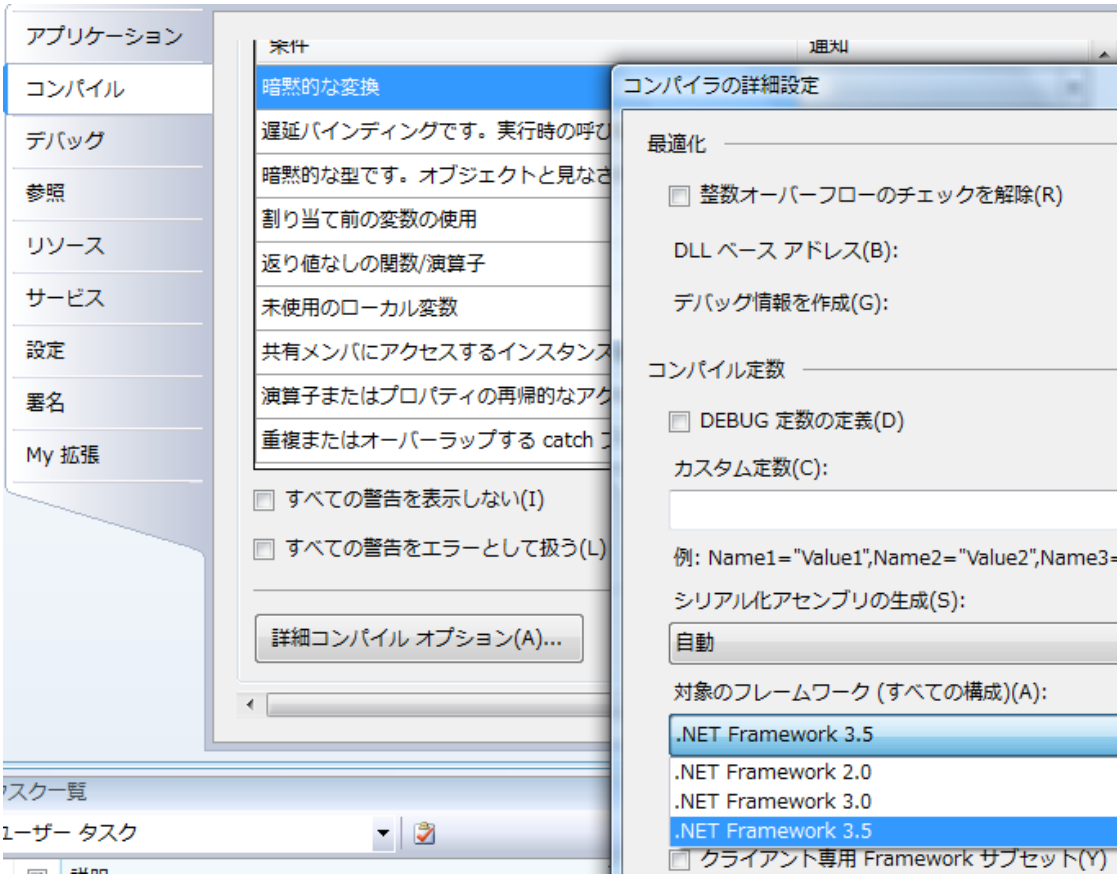
トラブルシューティング

AutoCAD.NET アセンブリ(AcDbMgd と AcMdg)　は最新バージョンの.NET Framework が必要です

アップグレードウィザード実行後、AutoCAD.NET アセンブリを追加すると下記のメッセージが表示される場合があります。この警告は移行されたプロジェクトが NET framework 2.0 で作成されていますが、AutoCAD は 3.5 バージョンを使用しているからです。“はい”をクリックして警告を閉じて下さい。



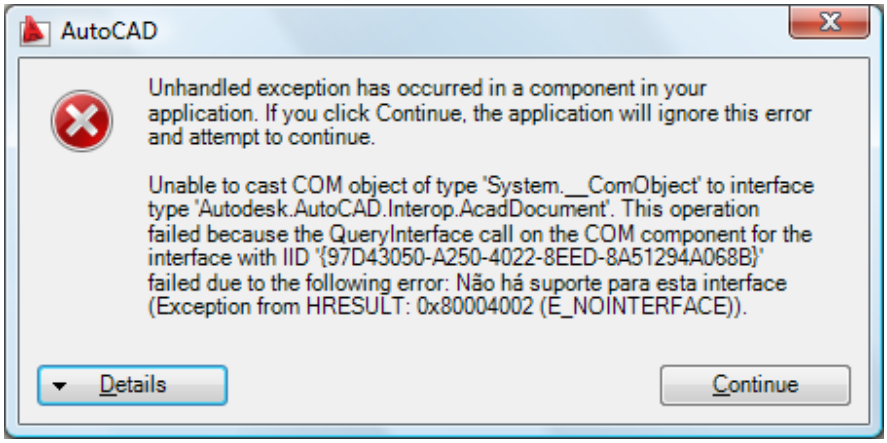
ソリューションエクスポーラでプロジェクト名を右クリックしてプロジェクトのプロパティにいきます。下記の様に“コンパイル”タブを選択し、「詳細コンパイルオプション」をクリックして“対象のフレーワーク”を“.NET Framework 3.5”に変更して OK をクリックします。Visual Basic はプロジェクトを保存、閉じて、再度開くように促します。



COM オブジェクト例外をキャストすることができません

下記の例外は AutoCAD COM オブジェクトを.NET で使用する場合はよく見られます。AutoCAD は互換性を周期的に保っています。もし、コードが メージャーバージョン R17 (2007/2008/2009)　の COM を使用してコンパイル済みで、AutoCAD　2010 (R18)で実行した場合は例外が発生します。似た様な例外は 32 ビット COM でコンパイルしたコードを 64 ビットの AutoCAD で実行した場合やその逆の場合にも見られます。

この例外を解決するにはプロジェクトを適切なバージョンの COM 参照でコンパイルすることです。

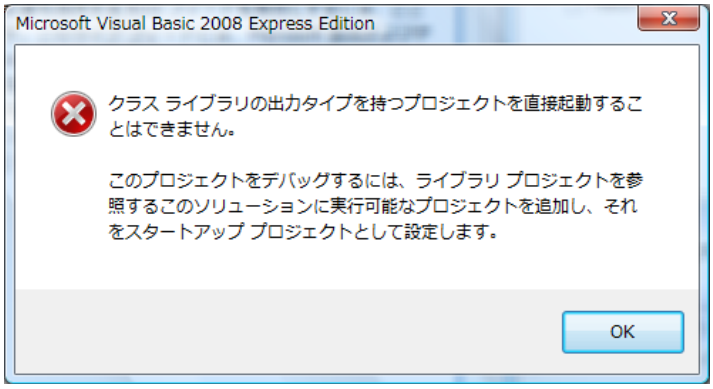


プロジェクトを直接開始することができません

「デバッグ」メニューで“デバッグ開始”を選択、または F5 キーを押した時、以下様なエラーが発生します。AutoCAD .NET アドインはクラス　ライブラリ (例：.dll ファイル) なので、それ自体で実行開始できません。　.exe ファイルが必要ですが、この場合 acad.exe がそれにあたります。

Express Edition 以外の Visual Studio では外部プログラムを自分のアプリケーションを実行する目的で設定できますが、Express エディションではこの設定はできません。この制限のためにプロジェクトをデバッグすることがでませが、.vbproj ファイルを直接編集することでデバッグを有効にすることが可能です。 “Enabling Debug on Visual Basic Express” 項目を参照するか、または VBA から VB6 変換ツールの“Post-Process VB Express Project ”機能を使用して下さい。

必要な変更に関しての情報については、次のブログをご覧下さい: http://through-the-interface.typepad.com/through_the_interface/2006/07/debugging_using.html



その他のリソース

Through the Interface ブログ- http://through-the-interface.typepad.com
Kean Walmsley の AutoCAD .NET に焦点を当てた数多くのサンプルを含んだブログ

AutoCAD.NET デベロッパガイド日本語版 - www.autodesk.co.jp/developautocad
.NET (C# と VB.NET) サンプルと相当する VBA コードを含んだ AutoCAD .NET のドキュメント

AutoCAD デベロッパセンター – www.autodesk.co.jp/developautocad
トレーニングマテリアル、プレゼン録画、 AutoCAD .NET Wizard 等

ADN DevTV: AutoCAD VBA to .NET Migration Basics –
http://download.autodesk.com/media/adn/VBA_Migration/DevTV_Recording.zip

ディスカッショングループ - http://discussion.autodesk.com/forums/category.jspa?categoryID=8

オートデスク　デベロッパ　ネットワーク– http://www.autodesk.co.jp
（「パートナー」－＞「 ADN」－＞「OTW」－＞「ADN」)

Visual Basic Express 日本語版- http://www.microsoft.com/japan/msdn/vstudio/Express/