ラッパーインストラクションサンプル

AxWrapperGen によって作成されたソリューションに含まれている Instructions の一例を翻訳しました。

TODO: Complete the code generated for VB6vsElasticLight／VB6vsElasticLight 用コードを完成させる。

0) Ensure that the name you have selected for the .NET project (ie the argument of /project command line switch) doesn't match the name or the namespace of the ActiveX control. If these name match, you will receive a large number of compilation errors. In this case, change the project name in the Application page of the My Project folder.

.NET プロジェクト用にあなたが選んだ名前（例　/project コマンドラインスイッチの引数）が ActiveX コントロールの名前または名前空間と一致しないことを確認してください。これらの名称が一致している場合、非常に多くのコンパイルエラーが発生するでしょう。この場合、My Project フォルダのアプリケーションのページでプロジェクト名を変更してください。

1) ensure that original control name is correct (this is the 1st argument of VB6Object attribute) this string must match the control string as seen in VB6 Object Browser

元のコントロール名が正しいことを確認してください。（これは VB6Object 属性の第 1 引数です。）この文字列は VB6 のオブジェクトブラウザに表示されるコントロールの文字列と一致しなければなりません。

2) specify an AlternateName value for the VB6Object attribute if adding an instance of the control by means of the Controls.Add method requires that you specify a different name Example: "MSComctlLib.TreeCtrl.2" for the TreeView control)

異なる名称を規定する必要がある Control.Add メソッドを使ってコントロールのインスタンスを追加する場合、VB6Object 属性のための AlternateName を規定してください。 例　TreeView コントロールに対する「MSComctlLib.TreeCtrl.2」

3) assign the IgnoreMembers property a pipe-delimited list of members to be ignored, if any. Example: "Negotiate|TransparentColor"

もしあれば、パイプで区切られた無視されるメンバのリストに IgnoreMembers プロパティを割り当ててください。 例「Negotiate|TransparentColor」

4) assign the TranslateProperties property a comma-delimited list of (oldname=namename) members. Example: "Bindings=DataBindings,MaskFormat=Format"

カンマ区切りメンバ（古い名前=新しい名前）リストに TranslateProperties プロパティを割り当ててください。 例「Bindings=DataBindings,MaskFormat=Format」

```
5) if control supports data-binding
        edit the "BoundValue" and "BoundPropertyChanged" regions to ensure
        that name and type of bound property is correct
   else
        remove the "Implements IVB6BoundControl" statement
```

```
        remove the "-- Data-binding support" region
```

NOTE: if the ActiveX control exposes its own DataXxxx, these elements are automatically removed

コントロールがデータバインディングをサポートする場合、「BoundValue」と「BoundPropertyChanged」リージョンについてバインドプロパティの名前と型が正しくなるように編集してください。サポートしない場合は「Implements IVB6BoundControl」ステートメントと「-- Data-binding support」リージョンを削除してください。 NOTE：ActiveX コントロールが自身の DataXxxx を公開している場合、これらの要素は自動的に削除されます。

6) if control does *not* support "classic" (VB3-style) drag-and-drop (e.g. DragIcon property and Drag method)

```
        remove the "Implements IClassicDragDrop" statement
        remove the "-- Classic drag-and-drop" region
```

コントロールが「旧式」（VB3 スタイル）のドラッグアンドドロップ（例　DragIcon プロパティと Drag メソッド）をサポートしない場合、「Implements IClassicDragDrop」ステートメントと「-- Classic drag-and-drop」リージョンを削除してください。

7) rename members whose name matches a VB.NET keyword or the name of another member. (For a COM component it is legal to expose a property and an event with same name, but this isn't allowed in .NET therefore one of the members must be renamed.) This step often requires that you modify the code in two or more points.

VB.NET のキーワードと同じ名称のメンバと他のメンバと名前が同じメンバの名前を変更してください。（COM コンポーネントについては同じ名前のプロパティやイベントを公開することは問題ありませんが、.NET では許されていないため、メンバのうちひとつは名前を変更しなければなりません。この作業は 2 箇所以上のコード変更が必要となることがあります。

Example: assume that the original COM component exposes an event named Error, which is a reserved VB.NET keyword.

例　元の COM コンポーネントが Error という VB.NET のキーワードとして予約されている名前のイベントを公開していると仮定してください。

This is the code that AxWrapperGen generates:

これは AxWrapperGen が生成するコードです。

```
        Public Shadows Event Error As VB6BeforePageBreakEventHandler

        Private Sub Control_Error(ByVal sender As Object, ByVal e As AxVSFlex8L. _
        _IVSFlexGridEvents_ErrorEvent) Handles MyBase.Error
                VB6BeforePageBreakEventDispatcher.Raise(Me, "Error", e.errorCode, _
                e.showMsgBox)
        End Sub
```

You must rename this event to something else, for example Error6. You must edit both the Event statement and the code that invokes the VB6BeforePageBreakEventDispatcher.Raise method:

このイベントは、例えば Error6 のように、他のイベントに名前を変更しなければなりません。また、イベントステートメントと VB6BeforePageBreakEventDispatcher.Raise メソッドを呼び出すコードの両方を編集しなければなりません。

```
Public Shadows Event Error6 As VB6BeforePageBreakEventHandler

Private Sub Control_Error(ByVal sender As Object, ByVal e As AxVSFlex8L. _
_IVSFlexGridEvents_ErrorEvent) Handles MyBase.Error
        VB6BeforePageBreakEventDispatcher.Raise(Me, "Error6", e.errorCode, _
        e.showMsgBox)
End Sub
```

Next, you must inform VB Migration Partner that you've renamed the event. You do it by setting the TranslateEvents property of the VB6Object attribute that precedes the Class statement:

次に、イベントの名前を変更したことを VB Migration Partner に知らせなければなりません。これは、Class ステートメントの前に置かれた VB6Object 属性の TranslateEvents プロパティを設定することで行います。

```
TranslateEvents:="Error=Error6"
```

8) Check whether there are two or more properties with same name but different argument syntax. These properties may be the result of a combination of get/set methods with different syntax. VS.NET prevents you from displaying a form at design time if the form contains such a control. You should delete all the overloads except one or (preferably) merge the overloads in a single property that takes optional arguments.

同じ名前であっても引数構文が異なるプロパティが二つ以上あるかどうか確認してください。これらのプロパティは異なる構文を持つ get/set メソッドの組み合わせの結果かもしれません。VS.NET は、フォームがそのようなコントロールを含む場合、デザイン用フォームを表示できないようにします。ユーザーは 1 個を除いてすべての多重定義を削除するか、多重定義を任意の引数を持つ一つのプロパティに統合すべきです。

Example: when applied to FlexGrid control, AxWrapperGen generates the following duplicated ArchiveInfo properties:

例　FlexGrid コントロールに適用する場合、AxWrapperGen は次のように複製された ArchiveInfo プロパティを生成します。

```
Public ReadOnly Property ArchiveInfo(ByVal arcFileName As String, ByVal _
infoType As VSFlex8L.ArchiveInfoSettings) As Object
        Get
                Return MyBase.get_ArchiveInfo(arcFileName, infoType)
        End Get
End Property

Public ReadOnly Property ArchiveInfo(ByVal arcFileName As String, ByVal _
infoType As VSFlex8L.ArchiveInfoSettings, ByVal index As Object) As Object
        Get
                Return MyBase.get_ArchiveInfo(arcFileName, infoType, index)
```

```
        End Get
    End Property
```

You can merge the two code blocks in a single property by making the 3rd argument Optional and writing the code that delegates to either get_ArchiveInfo method in the base class, depending on whether the optional argument has been omitted

ユーザーは、任意の引数を除外するかどうかに従って、第 3 の引数 Optional を作成し、ベースクラスの get_ArchiveInfo メソッドのどれかの代理となるコードを記述することで、二つのコードブロックを一つのプロパティに統合することができます。

```
    Public ReadOnly Property ArchiveInfo(ByVal arcFileName As String, ByVal _
    infoType As VSFlex8L.ArchiveInfoSettings, Optional ByVal index As Object _
    = Nothing) As Object
        Get
            If index Is Nothing Then
                Return MyBase.get_ArchiveInfo(arcFileName, infoType)
            Else
                Return MyBase.get_ArchiveInfo(arcFileName, infoType, _
                index)
            End If
        End Get
    End Property
```

9) check whether the wrapper class contains members that weren't in the original ActiveX control. For example, we have noticed that AxImp.exe changes the name of a member if the AxHost base class exposes a member with same name. For example, a property named EditMode might be converted into CtlEditMode, and Text might be converted into CtlText.

ラッパークラスが元の ActiveX コントロールにあったメンバを含むかどうか確認してください。例えば、私たちは AxHost のベースクラスが同じ名前のメンバを公開している場合、AxImp.exe がメンバの名前を変えることに気付いています。例えば、EditMode という名前のプロパティは CtlEditMode に変換されるかもしれませんし、Text が CtlText に変換されるかもしれません。

You can't change the names of these members (else the wrapper class would crash the Visual Studio .NET form designer), but you should edit the TranslateMembers property of the VB6Object attribute to specify which transformations might be necessary when converting the VB6 code that accesses those members, for example:

ユーザーはこれらのメンバの名前を変えることはできません(さもないとラッパークラスは Visual Studio .NET のフォームデザイナーを滅茶苦茶にしてしまいます。)が、それらのメンバにアクセスする VB6 コードを変換するときにどの変換が必要となるかを決める VB6Object 属性の TranslateMembers プロパティを編集するべきです。例えば、

```
    ... TranslateMembers:="Text=CtlText,EditMode=CtlEditMode", ...
```

Candidate members for this class are: CtlEnabled

このクラスに対するメンバ候補は CtlEnabled です。

10) check all the properties in the "--- Other Properties" region. Properties in this region might require manual fixing for the following two reasons:

「--- Other Properties」リージョンのすべてのプロパティを確認してください。このリージョンのプロパティは次の二つの理由によって手修正を必要とするかもしれません。

- they take or return values that are affected by the current ScaleMode それらは現行の ScaleMode によって影響を受けている値を取ったり返したりします。 Hint: use methods in the VB6Utils class to convert among different scale modes ヒント:異なるスケールモード間で変換するには VB6Utils クラスのメソッドを使用してください。
- they might be read-only properties or runtime-only properties that shouldn't appear in Visual Studio's property grid. それらは Visual Studio のプロパティウインドウに表示されるべきではない読み取り専用プロパティまたは実行時専用プロパティかもしれません。 Hint: prefix the Property definition with the following attributes: ヒント:次の属性をプロパティ定義の前に記述してください。
- <Browsable(False)> _
-
- <DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)> _
  Public Property .....

The Browsable attribute hides the property in the Property window; the DesignerSerializationVisibility attribute tells Visual Studio that the property's value must not be initialized in the *.Designer.vb file.

Browsable 属性はプロパティウインドウのプロパティを隠します。DesignerSerializationVisibility 属性はプロパティの値が *.Designer.vb ファイルで初期化されてはならないことを Visual Studio に伝えます。

11) check that no property is flagged with the following remark:

次のコメントによって警告されているプロパティがないことを確認します。

> TODO: .NET properties cannot have ByRef parameters. Manually fix this code as necessary. ／.NET プロパティは ByRef パラメータを持つことができません。このコードを必要に応じて手修正してください。

Each ByRef parameter in a property causes a compilation error. You can fix these errors in two ways:

プロパティの ByRef パラメータはコンパイルエラーの原因となります。これらのエラーは二つの方法で修正することができます。

- if the OCX documentation specifies that no value is actually returned through the parameter, you can safely each ByRef keyword into ByVal
  OCX の仕様書に、実際にはこのパラメータを通して戻される値がないことが明記されている場合、ByRef キーワードを ByVal キーワードに置き換えてかまいません。
- else, you need to preserve the ByRef semantics; in this case you can only split the property into two methods, get_propname and set_propname (you will also need to modify code in the client application.)
  さもなければ、ユーザーは ByRef 動作を維持する必要があります。この場合、ユーザーはプロパティを get_propname と set_propname の二つのメソッドに分けることしかできません。(また、ユーザーはクライアントアプリケーションのコードを修正する必要があります。)

12) once the code has no compilation errors, compile the solution in Release mode and then copy the following files to VB Migration Partner's installation folder (or a folder pointed to by an AddLibraryPath pragma)

コンパイルエラーが無くなった時点で、リリースモードでソリューションをコンパイルし、次のファイルを VB Migration Partner のインストールフォルダ（もしくは、AddLibraryPath プラグマを使って指定されたフォルダ）にコピーしてください。

AxVSLight6.dll, vsElasticLightLib.dll, VSLight6.dll You can find all these files in the VSLight6¥bin¥Release folder.

これらのファイルはすべて VSLight6¥bin¥Release folder.に見つけることができます。

13) you should be now able to migrate a VB6 form that hosts the ActiveX controls and obtain a VB.NET project that has no compilation error. However, you might still experience runtime errors.

この時点でコンパイルエラーなく ActiveX コントロールを含む VB6 フォームを変換して VB.NET プロジェクトを作成することができるはずです。しかし、まだ実行時エラーは発生するかもしれません。

A common error with ActiveX wrappers is that VB Migration Partner mistakenly generates one or more properties in the *.Designer.vb that shouldn't be there. For example, we have met this problem with a property named LicenseKey. Such spurious properties typically cause an exception when you display the form at design-time or when you run the VB.NET project. In such cases you must prevent VB Migration Partner from generating a property with that name. You achieve this by additing the name of the property to the IgnoreMembers property of the VB6Object attribute (see point 3).

ActiveX ラッパー共通のエラーは VB Migration Partner が、あるべきではない*.Designer.vb のプロパティを一つ以上誤って変換するためです。例えば、私たちは LicenseKey という名前のプロパティの問題に遭遇しました。そのような偽のプロパティが通常、デザイン用フォームを表示する際や VB.NET プロジェクトを実行する際のエラーの原因となります。そのような場合、ユーザーは VB Migration Partner がその名前のプロパティを生成しないようにしなければなりません。ユーザーは VB6Object 属性の IgnoreMembers プロパティ（3 参照）にそのプロパティの名前を追加することでこれを行うことができます。

14) If you notice that the ActiveX works well inside the VB.NET project except it doesn't retain the size (i.e. Width and Height properties), you should uncomment a group of statements in the ParseProperties method of the _Support class, as explained by the TODO comment.

ActiveX が、サイズ（例　Width プロパティと Height プロパティ）を保持できないこと以外は VB.NET プロジェクト内部でうまく動作することが確認できたら、TODO コメントによって説明されているように、_Support クラスの ParseProperties メソッドのステートメントを非コメント化するべきです。