Inside LateBindingApi.Excel

Was LateBindingApi.Excel im Kern tut ist LateBinding Aufrufe via .NET Reflection zu einem COM Server zu typensicheren Code zu mappen. Das Verfahren ist ähnlich der Generierung eines WebServiceClientProxies basierend auf WSDL. Der ClientProxy verbirgt die Web Requests dabei stellt reguläre Aufrufe zur Verfügung. LateBindingApi.Excel tut im Prinzip das gleiche jedoch derbirgt es LateBinding Aufrufe um das bekannte Excel Objekt Modell zur Verfügung zustellen.

Workarrounds:

C# unterstützt keine optionalen Parameter
- LateBindingApi generiert jeweils eine minimale und eine maximale Variante als Funktionsüberladung
Eine Funktion mit 2 Parametern und 3 optionalen Parametern hat demzufolge eine Funktion mit 2 Parametern und eine Überladung mit 5 Parametern. Für nicht verwendete Parameter nutzen sie System.Reflection.Missing.Value.

C# unterstützt keine Properties mit Parametern
- Properties mit Paramter werden zu Funktionen umgewandelt. Im alle eines Properties mit optionalen Parameten ebenso da es nicht möglich ist ein Property mit einer Funktion zu überladen.

C# unterstützt keinen Variant Datentypen
- Variant Datentypen werden durch Überladungen ihrer möglichen Typen ersetzt
Variante Indexer in Collections werden zu string und int gewandelt.

C# unterstützt keine Properties ihres einkapselnden Typs
- In C# ist es nicht möglich das beispielsweise eine Klasse MyApplication ein Property MyApplication MyApplication besitzt. Aus diesem Grund führt LateBindingApi.Excel das Präfix Xl für seine Wrapper Klassen.

C# unterstützt keine gleichnamigen Events mit anderen Methoden
Manche Klassen die Events zur Verfügung stellen führen Methoden die dem Namen eines exportierten Events gleich ist. Dies ist in C# nicht zulässig, XlLateBinding ergänzt in diesem Fall den Namen das Events mit „Event" so wird beispielsweise „Activate" zu „ActivateEvent".

COM Reference Handling

Wir haben 2 verschiedene Typen von Properties in Excel, Scalare Properties und COM Reference Properties. Scalare Properties werden von COMInterop kopiert und bedürfen keiner zusätzlichen Behandlung. COM Reference Properties werden im Speicher des COM Servers gehalten und lediglich einen Proxy darauf zurückgegeben. Für jeden erstellten Proxy muss der Client mittels einer speziellen Funktion signalisieren das er diesen nicht mehr benötigt da sonst Memory Leaks auftreten und/oder der COM Server nach beenden nicht aus der Prozessliste entfernt wird. Dieser Umstand führt in der Praxis oft zu Fehlern bzw. bläht den Code auf. LateBindingApi.Excel verwaltet daher die COM Referencen für Sie. Dabei haben Sie 3 Möglichkeiten der Verwaltung.

1.) Sie überlassen alles LateBindingApi.Excel

- In diesem Fall werden alle COM Referenzen auf einem internen Stapel abgelegt. Sobald Sie Excel beenden und Disopse() aufrufen werden die Referenzen freigegeben. Dies ist der einfachste und bequemste Fall, wenn sie jedoch umfangreich an Excel arbeiten können sehr viele erzeugte COM Referenzen die Performance beeinträchtigen da der Stapel immer grösser wird und auch der Excel COM Server den Speicher weiter mit den erzeugten Objekten belegt auch wenn Sie diese nicht mehr brauchen.

2.) Sie geben erzeugte Referenzen selbst frei.
- Jedes Klasse in LateBindingApi.Excel verfügt über die Methode ReleaseCOMReference() die Sie dafür nutzen können. Dies entspricht der gleichen Verhaltensweise wie die Primary Interop Assemblies.

3.) Sie nutzen das hierarchische Verwaltungsfeature in LateBindingApi.Excel. Betrachten Sie folgendes Listing:

Sie können sehen wie Referenzen auf ein Workbook, ein Worksheet und eine Range erstellt werden, die Range Referenz wird dabei nicht explizit reserviert. Der auf Aufruf von Dispose() für das workbook gibt alle Referenzen bis auf excelApplication frei. Warum? Wenn sie Dispose() für ein Objekt aufrufen wird alle über das Objekt erzeugten Objekte wiederum Dispose() aufgerufen. Da workSheet über workbook erstellt wurde wird also workSheet.Dispose() aufgerufen. Da die benutzte Range über Worksheet erstellt wird ebenfalls wieder Dispose() aufgerufen. Auf diese Weise können sie Code schreiben der nur einerseits die erzeugten COM Referenzen im Speicher klein hält ohne den Code mit Release Aufrufen aufzublähen.

```
XlApplication excelApplication = new XlApplication();
XlWorkbook workBook = excelApplication.Workbooks.Add();
XlWorksheet workSheet = workBook.Worksheets[1];
workSheet.Range("$A1").Text = "Hello";
workBook.Dispose();
```

Ereignisse:

Ereignisse werden in LateBindingApi.Excel durch COM Connection Points realisiert. Dies funktioniert jedoch erst ab Excel 2000. Der Support für Ereignisse ist per Voreinstellung ausgeschaltet. Nutzen Sie XlLateBindingApiSettings.EventsEnabled um den Event Support zu aktivieren. Folgende Typen unterstützen Ereignisse:

XlApplication
XlWorkbook
XlWorksheet
XlChart
XlCommandBars
XlCommandBarButton
XlCommandBarPopup
XlCommandBarComboBox

Das Beispielprojekt example8.csproj. demonstriert die Nutzung von Ereignissen.

Einstellungsmöglichkeiten

XlLateBindingApiSettings ist eine statische Klasse mit folgenden Properties.

Version FrameworkVersion – ReadOnly: Die aktuelle Framework Version

CultureInfo XlThreadCulture – Die verwendete Kultur bei allen LateBinding Aufrufen

bool EventsEnabled – Legt fest ob der Ereignisse ausgelöst werden sollen.

bool MessageFilterEnabled – Implementiert einen MessageFilter der verhindert das Excel Dialoge anzeigt insbesondere den OLE TASK WAITING Dialog wenn Excel ausgelastet ist. Da DisplayAlerts hier wirkungslos ist ein nützliches Feature.

Konzept von gekapselten LateBinding Aufrufen genau betrachtet

Der beste Weg dies zu erklären ist ein Blick in den Source Code des Property Visible der Klasse XlApplication.

```
public bool Visible
{
      get
      {
               object returnValue  = InstanceType.InvokeMember("Visible",
BindingFlags.GetProperty, null, ComReference, null,
XlLateBindingApiSettings.XlThreadCulture);
               return (bool)returnValue;
       }
       set
       {
               object[] parameter = new object[1];
               parameter[0] = value;
               InstanceType.InvokeMember("Visible", BindingFlags.SetProperty, null,
ComReference, parameter, XlLateBindingApiSettings.XlThreadCulture);
       }
}
```

Folgende 3 Punkte sind hierbei interessant:

1.) Type _InstanceType ist ein privates Feld, erstellt in der Methode
2.) object _ComReference ist ein privates Feld und der eigentlich COM Proxy zu Excel. Das Feld wird ebenfalls in der Methode CreateCOMReference erstellt.
3.) XlLateBindingApiSettings.XlThreadCulture ist die übergebende Kultur. Weitere Information dazu finden Sie unter „Einstellungsmöglichkeiten“

Klassen und Interfaces Überblick:

LateBindingApi.Excel kennt 2 verschiedene Arten von Klassen. Erstellbare und nicht erstellbare, dies ist dem Prinzip von Excel bzw. COM geschuldet.

Nicht erstellbare Klassen: Bei der grossen
Mehrheit aller Klassen können Sie keine Instanzen direkt erstellen sondern erstellen Sie über einen andere Klasse.
Diese Klassen implementieren in LateBindingApi.Excel das Interface IXlNonCreatable.

Erstellbare Klassen: Bei erstellbaren Klassen können sie eigenständige Instanzen davon erstellen. Diese Klassen implementieren in LateBindingApi.Excel das Interface IXlCreatable.

Klassen die Events zur Verfügung stellen implementieren ausserdem das Interface IxlEventBinding.

```
// Any object in the Framework implements this root interface
// the methods handle the Com Reference manage mechansim
public interface IXlBase : IDisposable
{
  object COMReference { get; }
  IXlBase ParentReference { get; set; }
  void ReleaseCOMReference();
  void RemoveChildReference(IXlBase comReference);
  void ReleaseChildReferences();
}

// Createble objects in the Framework implements this interface..
public interface IXlCreatable : IxlBase
{
   void CreateCOMReference(string progId);
}

// Non creatable objects in the Framework implements this interface.
public interface IXlNonCreatable : IXlBase
{
}

// Objects there raising events implements this interface.
public interface IXlEventBinding
{
   void SetupEventBinding();
   void RemoveEventBinding();
}
```

Ressourcen:

Using early binding and late binding in Automation
http://support.microsoft.com/kb/245115/en-us

Binding for Office automation servers with Visual C# .NET
http://support.microsoft.com/kb/302902/en-us

Receiving Events from late-bound COM servers
http://www.codeproject.com/KB/cs/zetalatebindingcomevents.aspx

Understanding COM Event Handling
http://www.codeproject.com/kb/com/TEventHandler.aspx