Entwicklung für Office/Excel aus .NET – Ein kurzer Überblick

Begriffsdefinition

COM Type Libraries
COM Type Libraries enthalten spezifische Informationen über COM Komponenten, z.B. welche Objekte, Aufzählungen, etc. sie zur Verfügung stellen. Dabei bekommt jedes Objekt sowie jedes Element(Property, Method) des Objekts eine eindeutigen numerischen Identifizierer(Dispatch Id) zugewiesen. Entwicklungsumgebungen nutzen bsp. Type Libraries um IntelliSense innerhalb des Code Editors anzubieten.

Early Binding
Early Bind Aufrufe werden anhand der unter COM Type Libraries beschriebenen numerischen Identifizierer realisiert.
Die COM Komponente stellt auf Nachfrage einen sogenannten Funktionszeiger zur Verfügung. Der Funktionszeiger kann direkt mit Parameterübergabe aufgerufen werden. Der Aufrufer der COM Komponente fragt konkret nach einer Methode mit der jeweiligen ID(z.B. 0x00000103) und bekommt den besagten Funktionszeiger geliefert. In C# und VB.NET oder auch in VB6 sehen von diesem Vorgängen allerdings nichts. Die jeweilige Programmiersprache erledigt diese Vorgänge für sie, in C# kümmert sich COMInterop um diesen Service. Dadurch dass die jeweilgen Funktionszeiger zwischengespeichert und direkt aufgerufen werden zeichnet sich das Verfahren durch besonders gute Performance aus, da die jeweiligen Identifizierer in den jeweiligen Excel Versionen jedoch unterschiedlich sind, funktioniert dieser Aufruf nicht versionsunabhängig

Late Binding
LateBinding beschreibt eine Technik Objekte sowie ihre Properties und Funktionen einzig anhand ihres Namens ohne Funktionszeiger aufzurufen. Existiert kein Property oder Objekt mit dem angegebenen Namen scheitert der Aufruf. Die Performance dieser Aufrufe ist vergleichsweise schlechter als der direkte Aufruf über den Funktionszeiger da die COM Komponente jedesmal abgefragt werden muss ob sie ein Objekt, Methode oder Property mit diesem Namen zur Verfügung stellt. Es ergibt sich daraus das dieser Ansatz tatsächlich nicht an eine bestimmte Version gebunden ist. Es bedeutet nicht das eine COM Komponente auch in jeder Version gleichnamige Objekte, Methoden, etc. zur Verfügung stellt. Glücklicherweise ist MS Office in seinem Objektmodell für den programmatischen Zugriff sehr beständig, daher ist das LateBinding Verfahren für den Zugriff auf MS Office Produkte sehr beliebt. LateBinding bietet natürlich keinerlei Komfort zur Entwicklungszeit hinsichtlich IntelliSense oder ähnliches da man auf anonymen COM Proxies arbeitet und sämtliche Informationen zur Laufzeit ermittelt werden. Zur Kompilierzeit findet keine Typenprüfung oder Prüfung auf die richtige Anzahl Parameter bei Funktionen etc. statt was eine hohe Fehleranfälligkeit provoziert. Code für LateBinding ist von Sprache zu Sprache sehr verschieden, in C# ist er um einiges aufwändiger und schwieriger zu lesen.

Zugriffswege auf MS Excel in .NET

Primary Interop Assemblies
Ein Primary Interop Assembly auch bezeichnet als Interop Dll das mittels des .NET Konsolenprogramms TlbImp.exe aus einer COM Type Library generiert wird. Wenn sie in Visual Studio einen Verweis auf eine COM Type Library hinzufügen wird automatisch eine Interop Dll generiert und ins bin\debug Verzeichniss ihres Projekts verschoben. Der Zugriff erfolgt early-bind ist also beschränkt auf die Version der Type Library. COM Type Libraries exisiteren immer für eine spezifische Anwendungsversion. Im Code-Editor bekommen sie IntelliSense Unterstützung, zur Laufzeit könne sie die Objekte im Debugger jedoch nicht untersuchen.
Die in Excel häufig vorkommenden Variant Datenytpen werden zu Object konvertiert. Sie müssen durch die

Dokumentation ermitteln welche Typen Variant einehmen kann und welche Folgen dies hat z.B. bei Parametern, Rückgabewerte oder Properties. Mehr dazu weiter unten Object und der Variant Datentyp. Bei paralell installierten Excel Versionen bzw. Downgrades einer Excel Version auf dem Zielystem kann es dabei zu Typenkonflikten kommen. COM Entwickler kennen dieses Problem als ‚Dll Hell'.

VSTO
Ein spezielles Programmiermodell von Microsoft das auf dem Entwicklungs und Zielsystem installiert und registriert sein muss. Es ermöglicht ihnen einen erweiterten Zugriff und ein spezielles Sicherheitsmodell für den Zugriff und ist als .NET Pedant zu VBA konzipiert. Der Zugriff auf Excel via VSTO ist eingeschränkt versionslos und nur als Addin oder Vorlage möglich. VSTO exisitert selbst in mehreren Versionen und arbeitet jeweils nur mit bestimmten .NET und Excel Versionen. Darüber hinaus existieren einige Deployment Erschwernisse, so muss Excel während der VSTO Installation und Registrierung auf dem Zielsystem bereits installiert sein und darf sich in der Folge in der Version auch nicht ändern, selbst wenn die VSTO Version die geänderte Version unterstützt. (Es ist zu vermuten das VSTO die Interfaces für den Zugriff auf Excel während der Registrierung dynamisch generiert und auf diese Weise mehrere Versionen unterstützt.)
Variant Datenytpen werden ebenfalls zu Object konvertiert mit den oben beschriebenen Konsequenzen.
http://msdn.microsoft.com/en-us/vsto/default.aspx (english)
http://blogs.msdn.com/b/jensha/archive/2010/04 /28/vsto-versionen-erkl-rt.aspx (german)

LateBinding und LateBinding Wrapper
Es gibt bereits mehr oder weniger rudimentäre Ansätze LateBinding Zugriffe auf COM im allgemeinen oder Office Produkte im speziellen in .NET komfortabler zu gestalten. 2 der gewichtigsten davon sind:

SafeComWrapper(http://www.codeproject.com/KB/COM/safecomwrapper.aspx) beschreibt eine Technik in C# um latebinded Aufrufe und Events zu kapseln. Im Ergebniss ist es möglich typensicheren Code zu schreiben. Die Objekte sind zur Laufzeit im Debugger jedoch nicht analysierbar. Die Wrapper für die jeweiligen COM Klassen müssen individuell von Hand erstellt und die Informationen der jeweiligen Dokumentation entnommen werden. Wenn sie nur sehr wenig Funktionalität von einer COM Komponente benötigen reicht ihnen dieser Weg möglicherweise aus.

Office for .Net(http://msofficefornet.codeplex.com) ist ein 1:1 Wrapper des Excel Objektmodells für den Latebinded Zugriff, geschrieben in Visual Basic.NET, die gleich mehrere Office Produkte untersützt, jedoch vor allem daran kränkelt im aktuellen Release 0.1.2 keine Events und nur die gängigsten Excel Objekte rudimentär zu unterstützen sowie keinen Workarround für Variants und Objects zu besitzen. Eine kurze Erläuterung finden sie hier.

Abschlussbemerkung: Manche Wrapper kapseln das Excelprogrammiermodell nicht 1:1 sondern versuchen ein eigenes verbessertes Objektmodell zur Verfügung zu stellen, jedoch sind diese Modelle nur gut für die Autoren dieser Modelle, im besonderen dann wenn ein unbeabsichtiges Verhalten analyisert werden muss.

Dynamic Objects in .NET 4
Dynamic Objects ähneln dem automatischen Dispatch Mechanismus bzw. dem Variant Datentyp von VB.NET. Dynamic Objects funktionieren auch mit COMInterop und ermöglichen die dynamische Typisierung eines Objekts. Der Compiler führt zur Kompilierungszeit keinerlei Prüfungen für ein Dynamic Object durch. Sie können beliebige Properties aufrufen, müssen jedoch sicherstellen das diese zur Laufzeit für das Objekt auch vorhanden sind. Dadurch exisiert im Folgeschluss auch kein Intellisense da keine TypenInformation vorhanden ist. Ebenso sind die Objekte zur Laufzeit nicht im Debugger analysierbar. Events werden unterstützt.Anmerkung: Auch wenn sie sich für die Verwendung von Interop Dll's oder VSTO entscheiden kann sich der Einsatz von Dynamic Objects für sie lohnen zum Beispiel um mit dem

Variant Datentypen umzugehen.
http://towardsnext.wordpress.com/2009/03 /16/dynamic-lookup-dynamic-type-c-40-part-1(english)
http://msdn.microsoft.com/en-us/library/dd264736.aspx(english)

Wissenswert: Verwaltung von COM Referenzen

Allen Zugriffsvarianten ist gemeinsam das sie selbst dafür Sorge tragen müssen die erstellten Objekte mittels einer speziellen Funktion vom COM Laufzeitsystem freizugeben. Tun sie dies nicht lebt die von ihnen erstellte Excel Instanz unter Umständen länger als sie möchten bzw. wird nicht ordentlich beendet. Ausserdem müssen Sie bedenken das jede COM Referenz auf eine erstellte Instanz innerhalb der COM Komponente verweist und damit Speicher belegt für dessen Freigabe Sie verantwotlich sind. Aufrufe mit mehrfachen Indirektionen verbieten sich dadurch.

Interop Dll Beispiel für die falsche Verwendung:

```
// falsch, häufiger fehler
string name = excel.ActiveWorkbook.Name;
```

Wir haben hier mit dem Zugriff auf ActiveWorkbook eine neuerliche COMReferenz erstellt die wir nicht speichern und somit auch nicht
freigeben können.

Interop Dll Beispiel für die richtige Verwendung:

```
// richtig
Workbook book = excel.ActiveWorkbook;
string name = book.Name;
Marshal.ReleaseComObject(book);
```

Anmerkung: Sie können die Garbage Collection veranlassen die Referenzen zu entsorgen Jedoch werden dann immer alle Referenzen anwendungsweit erstellten Referenzen entsorgt, in einfachen Anwendungsfällen sicherlich ausreichend, wenn sie mit mehreren Excel Instanzen gleichzeitig arbeiten jedoch möglicherweise problematisch.
Mit LateBinding.Excel müssen sie nicht auf die Art des Zugriffs achten, alle COM Referenzen werden von automatisch verwaltet.

Wissenswert: Object und der Variant Datentyp

Für den Zugriff auf eine COM Komponente sind die Informationen ihrer Type Library fast immer unerlässlich, jedoch sind diese Informationen teilweise sehr ungenau. Oftmals sind die Datentypen für Parameter nicht angegeben, gleiches gilt für Properties oder Rückgabewerte für Methoden. Die entsprechene Typenangabe fehlt oder sind als Object/Variant definiert. Dafür gibt es mehrere Gründe:

> Für Parameter gilt das dieser unterschiedlichen Typs sein darf, welcher Typ im Detail und die jeweiligen Konsequenzen müssen sie der Dokumentation entnehmen.
>
> Für Rückgabewerte aus Methoden und gilt nahezu das gleiche, je nach Parameter kann der Rückagebwert im Typ unterschiedlich ausfallen.

> Für Properties gilt das sie je nach Kontext der Klasse einen anderen Typ aufweisen, ausserdem ermöglicht er oftmals vereinfachte Zuweisung.
>
> Oftmals sind Properties vom Typ Object da sie von weiteren Type Libaries gemeinsam verwendet werden. TypeLibraries binden andere Type Libraries ein die damit nach aussen ein Teil der TypeLibrary werden. Beispielsweie binden alle Office Produkte binden die gemeinsame Office Type Library ein die Objekte für das GUI definiert. Die definierten Objekte besitzen ein Application Property. Da diese Library gemeinsam von vielen Office Applications benutzt wird ist dieses Property vom Typ Object.

In .NET setzt der Converter für Interop Dll's und damit auch VSTO und andere LateBindingWrapper für diese typenlosen Objekte den Typ Object ein. Die Deklaration dieser in der Type Library typenlosen Objekte in .NET zu Object bedeutet einen empfindlichen Verlust an Typensicherheit und Komfort. Sofern es sich dabei nicht um Wertetypen sondern Com Proxies handelt kommt die Problematik für nicht verwaltete COM Referenzen hinzu. Für den Zugriff aus C# ist zusätzlich problematisch das viele zulässige Merkmale in COM syntaktisch nicht unterstützt werden.

Anmerkung: LateBinding.Excel basiert auf einem eigenen Converter für Type Libraries, es findet kein Mapping zu Object statt und die Typensicherheit bleibt gewährleistet.

Performancevergleich der vorgestellten Zugriffswege

Fall 1: 5.000 Zellen einzeln nacheinander beschrieben mit "value"

Fall 2: 5.000 Zellen einzeln nacheinander beschrieben mit "value"
Änderung von Font, Numberformat und setzen von Border Rahmen mittels BorderArround

Fall 3: 10.000 Zellen einzeln nacheinander beschrieben mit "value"
Änderung von Font, Numberformat und hinzufügen eines Kommentars

Alle Testfälle wurden mit .Net 4 im ReleaseBuild ausgeführt.
Da VSTO nur innerhalb Excel verwendet werden kann und auf Primary Interop Assemblies aufsetzt ist es in der Tabelle nicht aufgeführt.
Leider fehlt Office for .Net im derzeitigen Release der notwendige Funkionsumfang so das hier nur Fall 1 getestet werden konnte.
Alle Tests wurden 20x ausgeführt und der gemittelte Wert in der Tabelle notiert

	Fall 1	Fall 2	Fall 3
Primary Interop Assembly	00:00:07.0500000	00:00:21.7000000	00:02:29.8000000
Dynamic Objects	00:00:19.4500000	00:00:33.3000000	00:03:23.6666666
Office for .Net	00:00:07		
LateBinding.Excel	00:00:07	00:00:22.8000000	00:02:37.6315789

Ein interessantes Ergebniss. Klarer Verlierer sind die Dynamic Objects in C#. Auffällig ist das LateBinding nicht unbedingt segnifikant langsamer ist als EarlyBinding, zumindest zeigen das die beiden LateBinding Libraries im Test. Sie finden die Quellcodes dieser Tests als Visual Studio 2010 Solutions im Ordner PerformanceTests wenn sie den Code downloaden.

Die Testfälle entsprechen keinem realen Szenario. Üblicherweise werden sie versuchen die Anzahl der Aufrufe bestmöglich zu reduzieren. Zum Beispiel über die Zuweisung von String Arrays oder Styles. Excel selbst ist in der Lage viele tausend Zellen in nur einer Sekunde zu beschreiben und formatieren, die Zahl der Aufrufe entscheidet über die Performance ihres Excel Codes. Die bestmögliche Art ihren Code dahingehend zu beschleunigen ist die Spreadsheet ML.