Vortrag

Integration von .Net-Komponenten in Access

Von
Philipp Stiefel

AEK 10
Nürnberg, Oktober 2007

Inhaltsverzeichnis

Einleitung

Dieser Vortrag behandelt, wie man das immer populärer werdende .Net-Framework nutzen kann, um Komponenten zu erstellen, die auch in einer Microsoft-Access-Anwendung verwendbar sind.

.Net? - Was ist das überhaupt?

Das .Net Framework ist eine Entwicklungs- und Laufzeitumgebung, die es ermöglicht, verschiedene Programmiersprachen und Bibliotheken nahtlos zusammen einzusetzen, um Windows-basierende Anwendungen zu erschaffen, welche einfach zu erstellen, zu verwalten und auszurollen sind und sich in andere vernetzte Systeme integrieren lassen.

(freie Übersetzung von http://www.microsoft.com/net/)

Wesentliche Punkte, die .Net ausmachen

- Common Language Runtime / Managed Code
  - Garbage Collection
  - Code Access Security
- Common Type System
- Framework Class Library

Subjektive Vorteile gegenüber Access

- Moderne Programmiersprachen
  - Vererbung
  - Exception Handling
- Die Framework Class Library
- (XCopy Deployment) (kein Vorteil im Kontext dieses Vortrages)

.Net-COM-Komponenten in Access verwenden (Part 1)

Warum .Net-Komponenten in Access nutzen?

- Erweiterung / Migration von Legacy-Anwendungen

- Einfachere Entwicklung bestimmter Funktionalität
- Interoperabilität von Anwendungen
- Wiederverwendung von existierendem Code

Wie kann man .Net aus Access nutzen?

- COM (Component Object Model) - ActiveX
- (WebServices) (sind nicht Thema dieses Vortrags)

COM-fähige .Net-Komponenten erstellen

Das .Net Framework ist die logische Weiterentwicklung der COM-Technologie. Daher ist Unterstützung von COM in das Framework integriert und dem SDK liegen zahlreiche Werkzeuge bei, die dem Entwickler bei der Erstellung von COM-Komponenten helfen können.

Mit Visual Studio

Folgende Schritte sind nötig:

- Neues Projekt erstellen („Class Library“)
- <Eigene Funktionalität implementieren>
- Projekteigenschaften konfigurieren:
  - Application –> Assembly Information -> Make assembly Com-Visible
  - Compile –> Register for COM Interop
- Kompilieren
- Fertig!

Achtung: Bei diesem Vorgehen werden ClassIds automatisch generiert. Besser die ClassIds eigener COM-Objekte selbst explizit festlegen! Also COM-Klassen entweder...

- auf Basis des Templates „ComClass“ erstellen (ab Visual Basic 2005 /.Net-Framework 2.0, nicht in der Express Edition),
- oder per Hand die Klassen mit dem Attribut `ComClass` (ab .Net 2.0., im VisualBasic-Namespace) versehen,

- oder die Klassen mit dem Attribut `GuidAttribute` (Net 1.0 / 1.1, Verzicht auf VisualBasic-Namespace) versehen.

Fazit: Erstellen von .Net-COM-Komponenten mit Visual Studio ist sehr einfach, aber es wird einiges hinter den Kulissen versteckt.

Ohne Visual Studio

d.h. ohne die Automatismen von Visual Studio

Ohne die Hilfe von Visual Studio müssen wir folgende Schritte ausführen:

- Neues Projekt erstellen
- <Eigene Funktionalität implementieren>
- Entweder für die Assembly oder für einzelne Klassen das Attribut `ComVisible` setzen.
- ClassIds aller Klassen, die `ComVisible` sind, definieren
- Kompilieren
- Mit `RegAsm.exe` die .Net-Assembly für COM registrieren (`/codebase` oder Assembly in den GAC!)
- Mit `Tlbexp.exe` die Type Library erstellen (auch mit `RegAsm` möglich)
- Mit `reglib.exe` die Type Library registrieren (auch mit `RegAsm` möglich)
- Fertig!

.Net-COM-Komponenten ohne die Hilfe von Visual Studio zu erstellen ist zwar etwas aufwendiger, aber wir lernen wichtige Details über die Zusammenhänge.

.Net-COM-Komponenten in Access verwenden (Part 2)

Type Library über Extras -> Verweise einbinden

Vorteile:

- Einfache Entwicklung mit Intellisense
- Events können verwendet werden (Variablendeklaration mit `WithEvents`)

Nachteile:

- Das Verweisdrama (Siehe FAQ 7.1)
- Type Library und .Net-DLL werden benötigt

.Net-COM.DLL mit Late Binding verwenden

Vorteile:

- Klassen und deren DLLs werden direkt über die Registry aufgelöst
  - Keine Aktualisierung von Verweispfaden auf anderen Rechnern nötig
  - Keine Gefahr von FAQ 7.1
  - .Net.Dll wird direkt verwendet. Keine Type Library nötig!

Nachteile:

- Events aus der Komponente können nicht verwendet werden
- Kein Objektkatalog, kein Intellisense

Achtung: Beim Late Binding dürfen die Typen in der CreateObject-Anweisung nicht über Dateiname.Klassename referenziert werden, sondern über Namespace.Klassenname!

Deployment von Access-Anwendungen mit .Net-Komponenten

Folgendes ist zu beachten:

- Die .Net-DLL und die Type Library müssen auf dem Zielrechner registriert werden
- Bei Verwendung des `/codebase`-switches für `RegAsm` wird der komplette Pfad zur DLL in der Registry eingetragen
- Ohne `/codebase`-switch muss die DLL in das Anwendungsverzeichnis (MSAccess.exe!) oder in den GAC (Strong Name!)

Undokumentiert: Controls mit .Net

Offiziell ist es nicht möglich, Steuerelemente mit .Net zu erzeugen, die analog zu den klassischen ActiveX-Controls verwendet werden können.

User Controls dennoch über COM verfügbar machen

- User Control in .Net erstellen und die Klasse als COM-Objekt vorbereiten.
- In der Registry, unterhalb der ClassId im Registry-Zweig `HKEY_CLASSES_ROOT\CLSID` noch der neue Schlüssel „Control“ angelegt werden. (Mit einer „`ComRegisterFunction`“ bzw. einer „`ComUnregisterFunction`“ kann das Erzeugen des Control-Keys automatisiert werden.)

Vorteile eines User-Controls gegenüber Formularen

- Kleinere Funktionseinheiten lassen sich modularisieren
- .Net-Forms können u.U. nicht vollständig in den Control-Flow einer Access-Anwendung integriert werden (Modale Formulare, Dialoge)
- Allgemein bessere Integration in Look&Feel der Host-Anwendung

Potentielle Probleme

Da COM-User-Controls nicht offiziell dokumentiert sind, sollte man auf Probleme vorbereitet sein. Z.B.

- Das Control resized sich von selbst auf die Größe, die es in der VS-IDE hatte.