Integration von .Net-Komponenten in Access

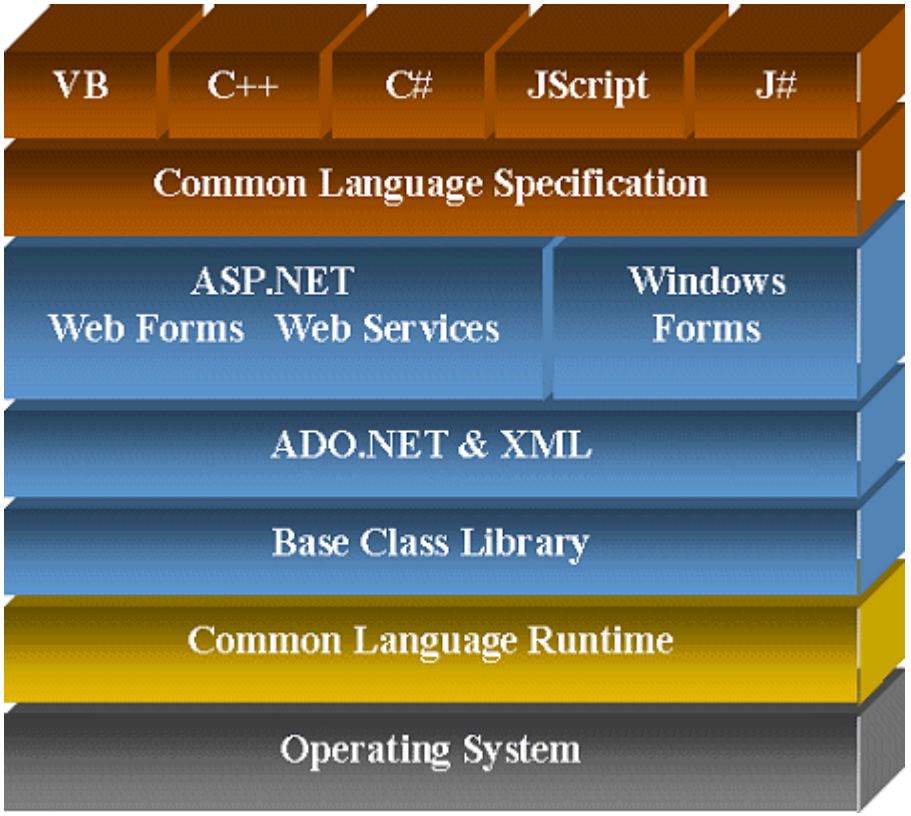
Philipp Stiefel
www.codekabinett.com

Gliederung

- .Net – Was ist das?
- .Net-COM-Komponenten in Access verwenden (Vorüberlegungen)
- COM-fähige .Net-Komponenten erstellen
- .Net-COM-Komponenten in Access verwenden (Umsetzung)
- Undokumentiert: Controls mit .Net
- Fazit
- Abspann (Literatur , F&A)

.NET – WAS IST DAS?

Das Microsoft .Net Framework



Was .Net ausmacht

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

Vorteile gegenüber Access

- Moderne Programmiersprachen
  - Vererbung
  - Exception Handling
- Die Framework Class Library

.NET-COM-KOMPONENTEN IN ACCESS VERWENDEN - VORÜBERLEGUNGEN

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?

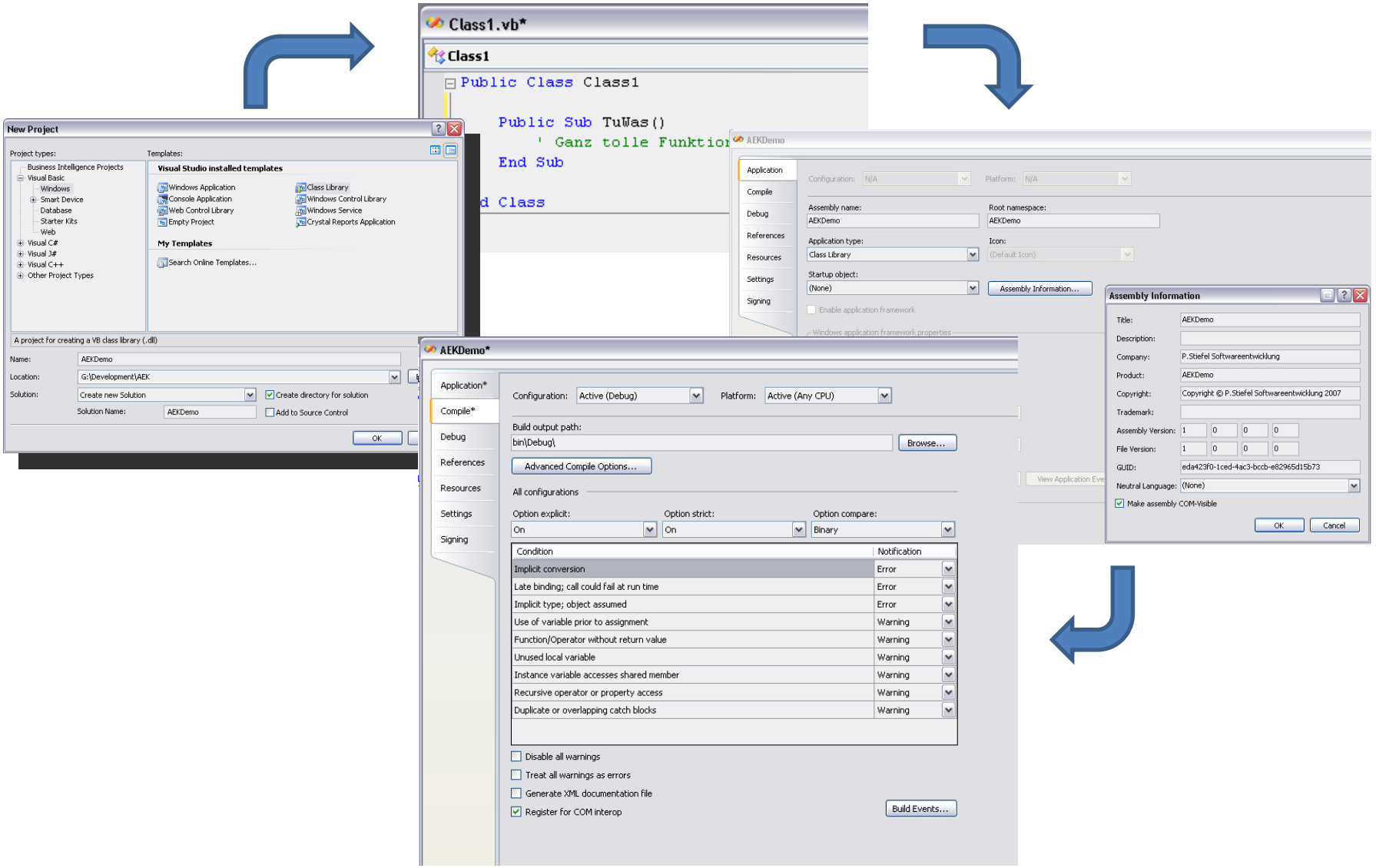
- COM (Component Object Model) - ActiveX
- (WebServices) (sind nicht Thema dieses Vortrags)

COM-FÄHIGE .NET-KOMPONENTEN ERSTELLEN

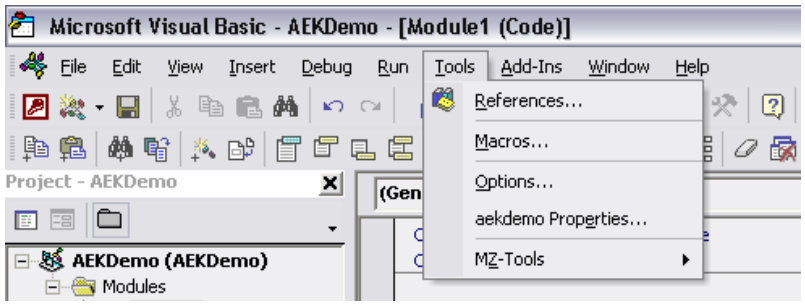
Mit Visual Studio

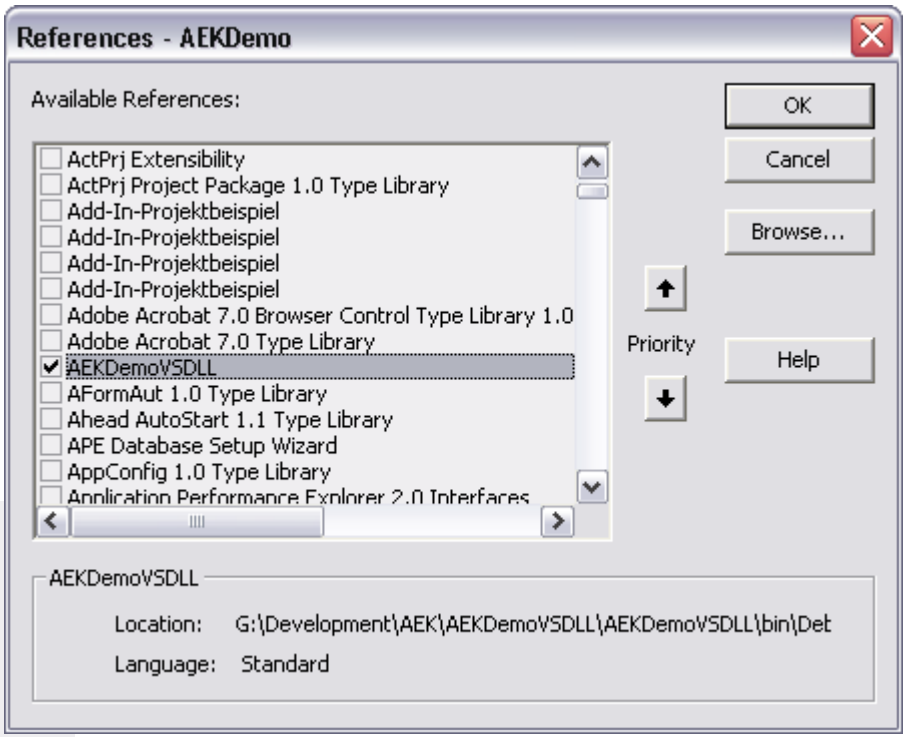
1. Neues Projekt erstellen („Class Library“)
2. <Eigene Funktionalität implementieren>
3. Projekteigenschaften konfigurieren:
    1. Application –> Assembly Information -> Make assembly Com-Visible
    2. Compile –> Register for COM Interop
4. Kompilieren

Demo – Visual Studio



Demo Access

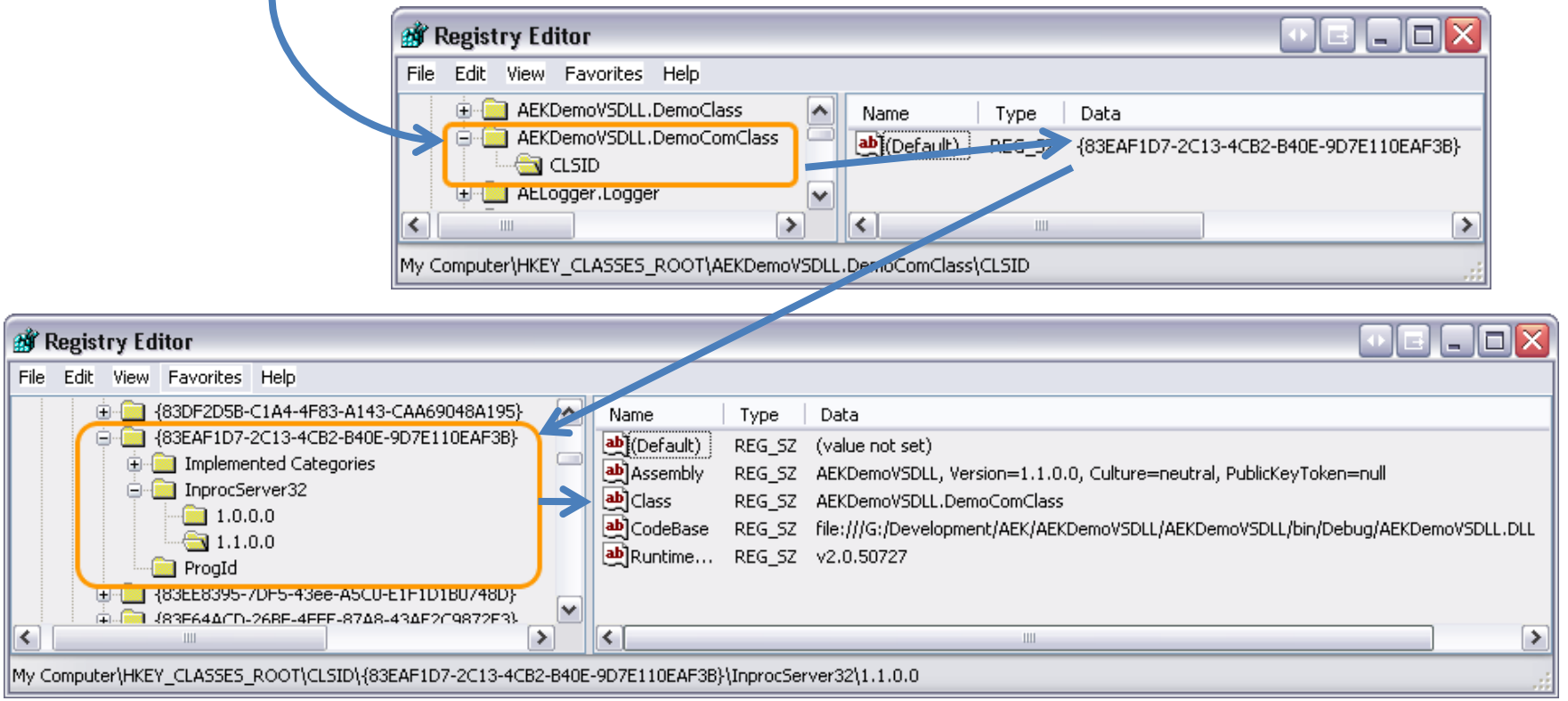





```
Public Sub Demo()

    Dim aekdemo As AEKDemoVSDLL.DemoComClass
    Set aekdemo = New AEKDemoVSDLL.DemoComClass

    aekdemo.ShowMessage "Ene mene muh!"

    Set aekdemo = Nothing

End Sub
```

ClassId - Bindeglied zwischen Klassennamen und Typdefinition

```
Public Sub Test()

    Dim o As AEKDemoVSDLL.DemoComClass
    Set o = New AEKDemoVSDLL.DemoComClass

End
```



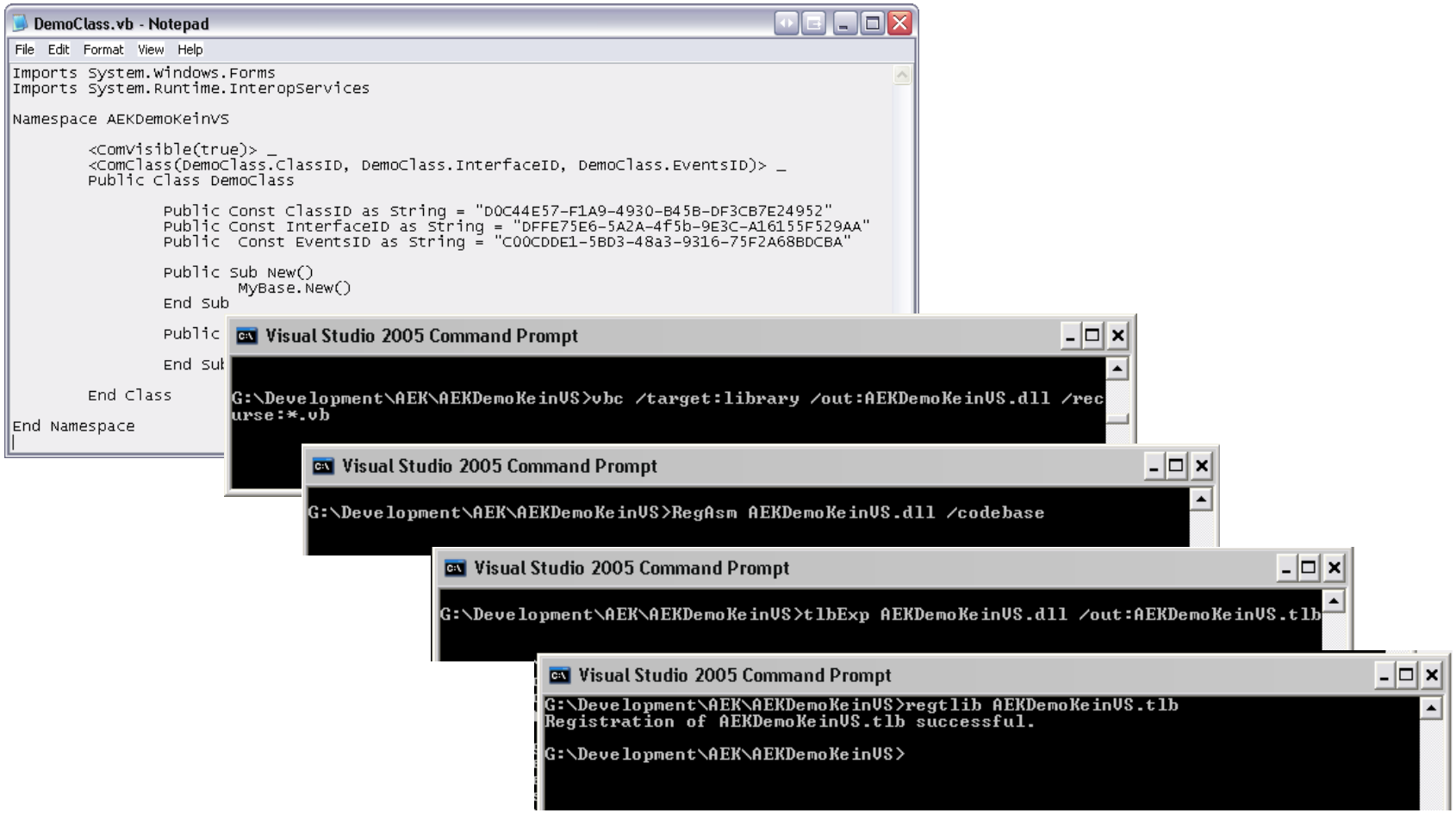
ClassIds explizit definieren

- Template ComClass (VB 2005)
- ComClassAttribute (.Net 2.0)
- GuidAttribute (.Net 1.0 / 1.1 / Nicht VB)

Ohne Visual Studio

1. Neues Projekt erstellen
2. <Eigene Funktionalität implementieren>
3. Entweder für die Assembly oder für einzelne Klassen das Attribut ComVisible setzen.
4. ClassIds aller Klassen, die ComVisible sind, definieren
5. Kompilieren
6. Mit RegAsm.exe die .Net-Assembly für COM registrieren (/codebase oder Assembly in den GAC!)
7. Mit Tlbexp.exe die Type Library erstellen (auch mit RegAsm möglich)
8. Mit regtlib.exe die Type Library registrieren (auch mit RegAsm möglich)

Demo – Ohne Visual Studio



.NET-COM-KOMPONENTEN IN ACCESS VERWENDEN - UMSETZUNG

Type Library als Verweis 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

.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

Demo – Late Binding?

```
Public Sub DemoLateBinding()

    Dim aek As Object
    Set dcc = CreateObject("AEKDemoVSDLL.DemoComClass")

    dcc.ShowMessage "Neue Nachricht!"

    Set dcc = Nothing

End Sub
```

Deployment von .Net-Komponenten

- 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

User Controls 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 den Schlüssel „Control“ anlegen

Demo – User Control

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

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

Demo – Ein Control aus der Praxis

FAZIT

Fazit aus den Praxiserfahrungen

- Praxistaugliche Technologie
- Vorsicht bei Verteilung und Aktualisierung!

ABSPANN

Quellen & Links

- .NET Framework Developer's Guide - Exposing .NET Framework Components to COM (ms-help://MS.MSDNQTR.v80.en/MS.MSDN.v80/MS.VisualStudio.v80.en/dv_fxinterop/html/e42a65f7-1e61-411f-b09a-aca1bbce24c6.htm)
- MSDN Library –> Win32 and COM Development -> Component Object Model
- Morgan Skinner - Exposing Windows Forms Controls as ActiveX controls (http://www.codeproject.com/cs/miscctrl/exposingdotnetcontrols.asp)
- Interop Forms Toolkit 2.0 (http://msdn2.microsoft.com/en-us/vbasic/bb419144.aspx)

Fragen?