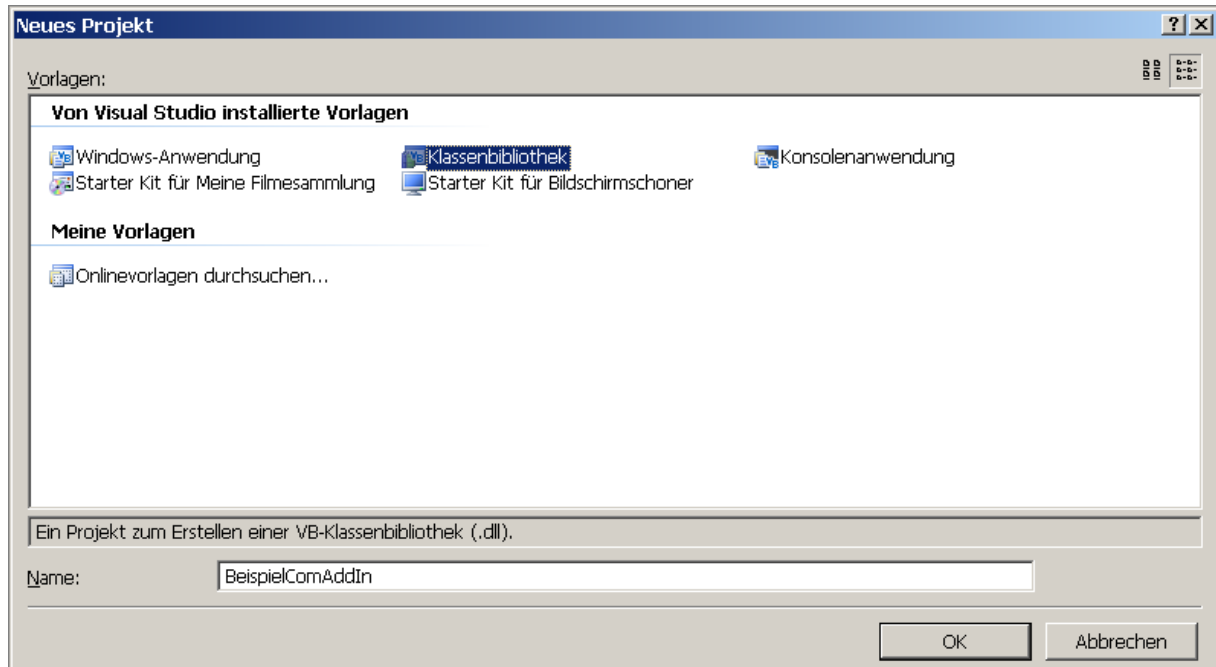


Erstellen eines Office-COM-Add-In mit Visual Basic 2005 Express Edition

Neues Projekt (Klassenbibliothek) erzeugen



Als Projektbezeichnung wählte ich BeispielComAddIn.

Add-In-Starter-Klasse erstellen

Die Klasse Namens AddInStarter wird im Verlauf dieser Beschreibung laufend erweitert, um die einzelnen Schritte bis zu einem COM-Add-In zu zeigen.

Klasse mit Public Function

```
Public Class AddInStarter

    Public Function Testfunktion() As String
        Testfunktion = "Dieser Text kommt von der Testfunktion"
    End Function

End Class
```

Kompilierung und Registrierung

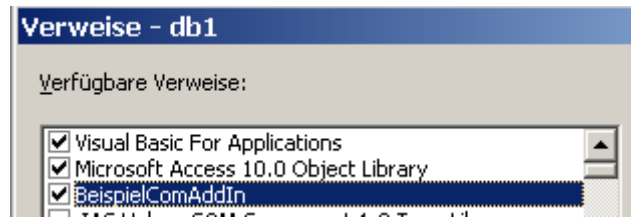
1. dll erstellen: Menü → Erstellen → BeispielComAddIn erstellen.
Im Projektordner unter \bin\Release liegen nun diese Dateien:
 - BeispielComAddIn.dll
 - BeispielComAddIn.pdb
 - BeispielComAddIn.xml
2. Für die COM-Registrierung fehlt noch die BeispielComAddIn.tlb. Diese Datei kann man bei der Registrierung mit regasm.exe als Parameter angeben.

```
regasm.exe BeispielComAddIn.dll /codebase /tlb:BeispielComAddIn.tlb
```

Die Parameter für regasm sind unter

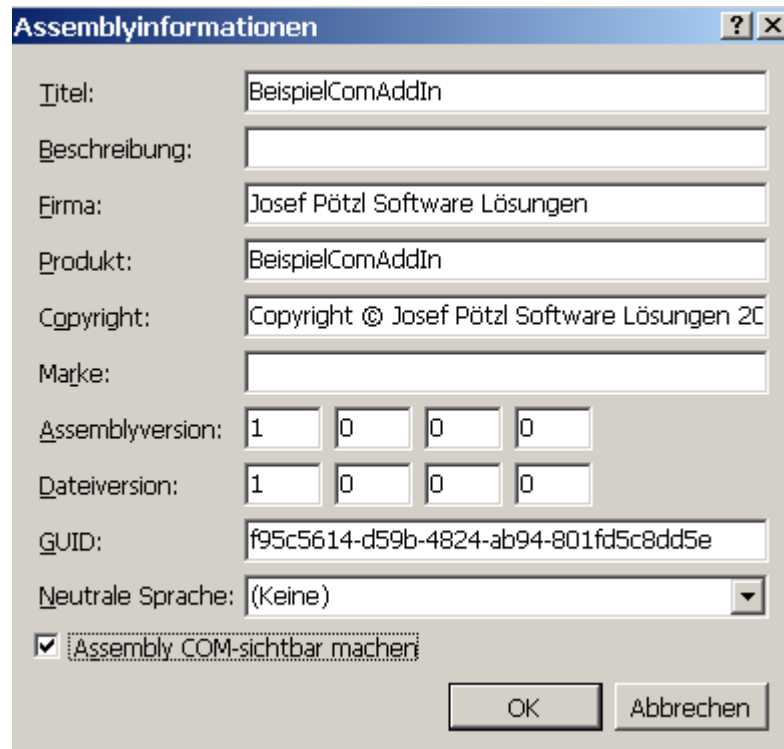
<http://msdn.microsoft.com/library/deu/default.asp?url=/library/DEU/cptools/html/cpgrfassemblyregistrationtoolregasmexe.asp> beschrieben.

Mit dem Schalter `/tlb:BeispielComAddIn.tlb` wird das BeispielComAddIn auch in den Verweisen angezeigt.



Betrachtet man diese Bibliothek im Objektkatalog, ist die Klasse `AddInStarter` nicht sichtbar. Damit diese Klasse sichtbar wird, ist das Assembly als COM-sichtbar einzustellen.

Menü Projekt → BeispielComAddIn-Eigenschaften → Anwendung → Assemblyinformationen



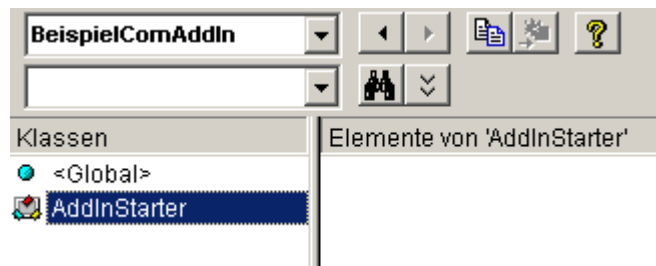
Alternativ kann statt dem gesamten Assembly nur die Klasse als COM-sichtbar markiert werden.

COM-sichtbare Klasse

```
<System.Runtime.InteropServices.ComVisible(True)> _  
Public Class AddInStarter  
    Public Function Testfunktion() As String  
        Testfunktion = "Dieser Text kommt von der Testfunktion"  
    End Function  
End Class
```

Hier darf man sich nicht von den angezeigten Klasseneinstellungen im Visual-Studio ablenken lassen. Dort wurde bei mir auch eine Klasse als COM-sichtbar angezeigt, obwohl ich `ComVisible(True)` nicht setzte und das Assembly nicht COM-sichtbar war. Im Objektkatalog wird es so angezeigt, wie es im Code eingestellt ist.

Nach erneutem Erstellen der dll und Registrierung mit regasm.exe ist die Klasse nun im Objektkatalog sichtbar. Es fehlt aber noch die public Function `Testfunktion()`.



Um auch diese anzuzeigen, muss die Klasse als COM-Klasse gekennzeichnet werden. (Klasseneigenschaft: COM-Klasse: True)

COM-Klasse

```
<System.Runtime.InteropServices.ComVisible(True)> _
<Microsoft.VisualBasic.ComClass(> _
Public Class AddInStarter
    Public Function Testfunktion() As String
        Testfunktion = "Dieser Text kommt von der Testfunktion"
    End Function
End Class
```

Diese Klasse ist nun bereits in VBA verwendbar.

```
Private Sub EarlyBindingAufruf()

    Dim ComObject As BeispielComAddIn.AddInStarter

    ComObject = New BeispielComAddIn.AddInStarter
    MsgBox(ComObject.Testfunktion)
    ComObject = Nothing

End Sub

Private Sub LateBindingAufruf()

    Dim ComObject As Object

    ComObject = CreateObject("BeispielComAddIn.AddInStarter")
    MsgBox(ComObject.Testfunktion)
    ComObject = Nothing

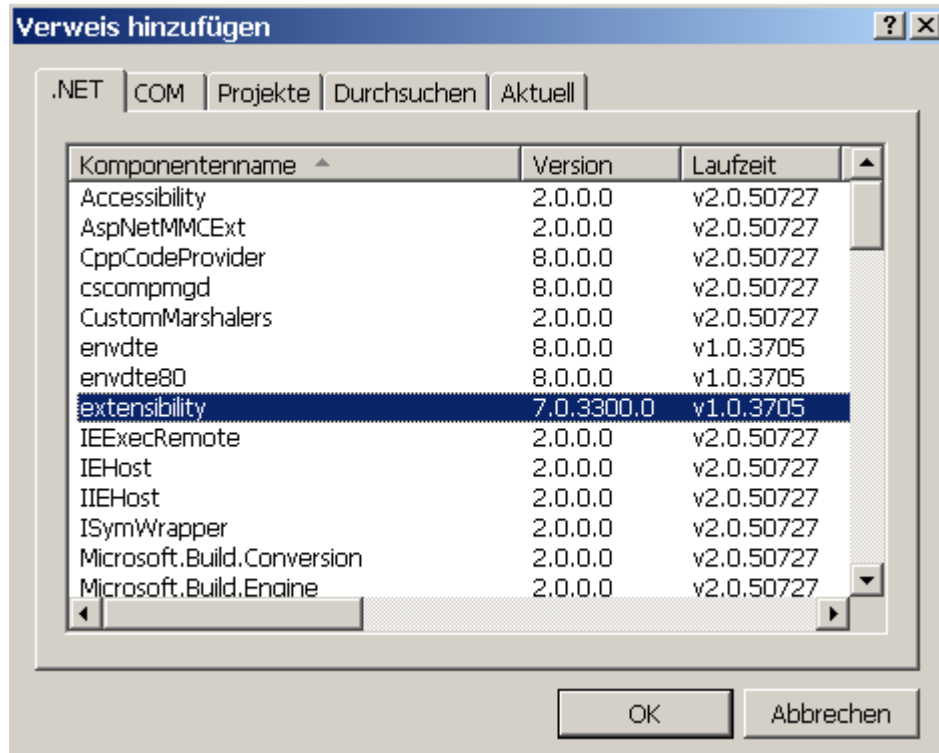
End Sub
```

COM-Add-In

Nun ist als letzter Schritt noch die Implementierung der Add-In-Funktionalität erforderlich. Dafür ist die Schnittstelle `IDTExtensibility2` zu implementieren.

Es fehlt in der Klasse noch `Implements Extensibility.IDTExtensibility2`

Damit dies erkannt wird, muss ein Verweis auf die `extensibility`-Bibliothek eingefügt werden.



Als nächstes sind die `IDTExtensibility2`-Methoden anzufügen.

```
<System.Runtime.InteropServices.ComVisible(True)> _
<Microsoft.VisualBasic.ComClass()> _
Public Class AddInStarter

    Implements Extensibility.IDTExtensibility2

    #Region "IDTExtensibility2"

        Public Sub OnBeginShutdown(ByRef custom As System.Array) _
            Implements Extensibility.IDTExtensibility2.OnBeginShutdown
            MsgBox("OnBeginShutdown")
        End Sub

    <usw.>
    #End Region

    Public Function Testfunktion() As String
        Testfunktion = "Dieser Text kommt von der Testfunktion"
    End Function

End Class
```

Ein wichtiger Schritt fehlt noch für die Funktionsfähigkeit des Add-Ins: es muss für die jeweilige Anwendung registriert werden.

Add-In registrieren

Für die Erkennung als Add-In sind Einträge in die Windows-Registry erforderlich. Das ist am einfachsten mit Registrierungs-Dateien durchzuführen.

Nutzung als VBA/VBE-AddIn:

VBAVBE.reg:

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\VBA\VBE\6.0\Addins\BeispielComAddIn.AddInStarter]
"LoadBehavior"=dword:00000000
"CommandLineSafe"=dword:00000000
"FriendlyName"="BeispielComAddIn-AddInStarter"
```

LoadBehavior stellt das Ladeverhalten ein.

0 ... manuelles Laden

1 ... Beim Start Laden

Nutzung als Access-Add-In

Access.reg:

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Office\Access\Addins\BeispielComAddIn.AddInStarter]
"LoadBehavior"=dword:00000003
"CommandLineSafe"=dword:00000000
"FriendlyName"="BeispielComAddIn-AddInStarter"
```

LoadBehavior 3 um das Add-In beim Access-Start zu laden.

Anm.:

Die Kennung `BeispielComAddIn.AddInStarter` funktioniert nur mit Standardnamespace `BeispielComAddIn`. Sollte ein erweiterter Namespace verwendet werden, so ist diese Kennung entsprechend anzupassen.

Wenn der Namespace z.B. `JoPoSoL.BeispielComAddIn` lautet, so ist `JoPoSoL.BeispielComAddIn.AddInStarter` zu verwenden.

Nun sind alle Voraussetzungen für das Schreiben eines COM-Add-In unter .net erfüllt.

Josef Pötzl, August 2007

Code-Beispiele

Die angeführten Klassen sind nur zur Veranschaulichung gedacht. Es fehlt eine vollständige Fehlerbehandlung sowie eine ausführliche Add-In-Steuerung.

Gerüst der COM-Add-In-Klasse

```
<System.Runtime.InteropServices.ComVisible(True)> _
<Microsoft.VisualBasic.ComClass()> _
Public Class AddInStarter

    Implements Extensibility.IDTExtensibility2

#Region "IDTExtensibility2"

    <System.Runtime.InteropServices.ComVisible(False)> _
    Public Sub OnBeginShutdown(ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnBeginShutdown
        MsgBox("OnBeginShutdown")
    End Sub

    <System.Runtime.InteropServices.ComVisible(False)> _
    Public Sub OnAddInsUpdate(ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnAddInsUpdate
        MsgBox("OnAddInsUpdate")
    End Sub

    <System.Runtime.InteropServices.ComVisible(False)> _
    Public Sub OnStartupComplete(ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnStartupComplete
        MsgBox("OnStartupComplete")
    End Sub

    <System.Runtime.InteropServices.ComVisible(False)> _
    Public Sub OnDisconnection(ByVal RemoveMode As _
        Extensibility.ext_DisconnectMode, _
        ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnDisconnection
        MsgBox("OnDisconnection")
    End Sub

    <System.Runtime.InteropServices.ComVisible(False)> _
    Public Sub OnConnection(ByVal application As Object, _
        ByVal connectMode As Extensibility.ext_ConnectMode, _
        ByVal addInInst As Object, _
        ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnConnection
        MsgBox("OnConnection")
    End Sub

#End Region

    Public Function Testfunktion() As String
        Testfunktion = "Dieser Text kommt von der Testfunktion"
    End Function

End Class
```

Erweiterte COM-Add-In-Klasse

Für diese Variante sind zusätzlich die Verweise auf Access und Office erforderlich.

```
Imports Microsoft
Imports System

<Runtime.InteropServices.ComVisible(True)> _
<Microsoft.VisualBasic.ComClass()> _
Public Class AddInStarter

    Implements Extensibility.IDTExtensibility2

    Private accessApplication As Access.Application
    Private WithEvents testFunktionCommandBarButton As _
        Office.Core.CommandBarButton
    Private WithEvents dataBaseInfoCommandBarButton As _
        Office.Core.CommandBarButton

#Region "IDTExtensibility2"

    Protected Sub OnBeginShutdown(ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnBeginShutdown
        'MsgBox("OnBeginShutdown")
        'Speicher freigeben
        Me.Dispose()
    End Sub

    Protected Sub OnAddInsUpdate(ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnAddInsUpdate
        'MsgBox("OnAddInsUpdate")
    End Sub

    Protected Sub OnStartupComplete(ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnStartupComplete
        If accessApplication IsNot Nothing Then
            If accessApplication.CurrentDb IsNot Nothing Then
                'Add-In nur laden, wenn eine Datenbank geöffnet ist.
                Me.addCommandBars()
            End If
        End If
    End Sub

    Protected Sub OnDisconnection(ByVal RemoveMode As _
        Extensibility.ext_DisconnectMode, _
        ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnDisconnection
        'MsgBox("OnDisconnection")
    End Sub

    Protected Sub OnConnection(ByVal application As Object, _
        ByVal connectMode As Extensibility.ext_ConnectMode, _
        ByVal addInInst As Object, _
        ByRef custom As System.Array) _
        Implements Extensibility.IDTExtensibility2.OnConnection
        If (TypeOf application Is Access.Application) Then
            accessApplication = CType(application, Access.Application)
        End If
    End Sub

#End Region


```

```

#Region "Dispose"

Protected Sub Dispose()

    '!!! Ohne GC.Collect
    '!!! schließt Access nicht und das Access-Fenster hängen.
    '!!! (Object = Nothing reicht nicht aus.)
    GC.Collect() ' !!!

    If accessApplication IsNot Nothing Then

        MsgBox(accessApplication.Name & " wird geschlossen")

        'eventuell um Marshal.ReleaseComObject erweitern.
        'in diesem BeispielComAddIn war es bei mir nicht notwendig.
        Try
            Runtime.InteropServices.Marshal.ReleaseComObject( _
                accessApplication)
        Finally
            accessApplication = Nothing
        End Try

        'If Not (testFunktionCommandBarButton Is Nothing) Then
        '    Try
        '        Runtime.InteropServices.Marshal.ReleaseComObject( _
        '            testFunktionCommandBarButton)
        '    Finally
        '        testFunktionCommandBarButton = Nothing
        '    End Try
        'End If

        'If dataBaseInfoCommandBarButton IsNot Nothing Then
        '    Try
        '        Runtime.InteropServices.Marshal.ReleaseComObject( _
        '            dataBaseInfoCommandBarButton)
        '    Finally
        '        dataBaseInfoCommandBarButton = Nothing
        '    End Try
        'End If

    End If

End Sub

#End Region

Public Function Testfunktion() As String
    Testfunktion = "Dieser Text kommt von der Testfunktion"
End Function

Private Sub addCommandBars()

    Dim officeCommandBar As Office.Core.CommandBar = _
        accessApplication.CommandBars.Add( _
            "BeispielComAddInMenu", _
            Office.Core.MsoBarPosition.msoBarTop, _
            Office.Core.MsoBarType.msoBarTypeNormal, _
            True)

    testFunktionCommandBarButton = CType(officeCommandBar.Controls.Add( _
        Office.Core.MsoControlType.msoControlButton, _
        1, , , True), Office.Core.CommandBarButton)

```

Erstellen eines Office-COM-Add-In mit Visual Basic 2005 Express Edition

```
With testFunktionCommandBarButton
    .DescriptionText = "Startet die Add-In-Funktion Testfunktion"
    .Style = Office.Core.MsoButtonStyle.msoButtonCaption
    .Caption = "Test"
End With

dataBaseInfoCommandBarButton = CType(officeCommandBar.Controls.Add( _
    Office.Core.MsoControlType.msoControlButton, _
    1, , , True), Office.Core.CommandBarButton)
With dataBaseInfoCommandBarButton
    .DescriptionText = _
        "Zeigt den Namen der geöffneten DB an."
    .Style = Office.Core.MsoButtonStyle.msoButtonCaption
    .Caption = "Currentdb.Name"
End With

officeCommandBar.Visible = True

[System.Runtime.InteropServices.Marshal.ReleaseComObject( _
    officeCommandBar)
officeCommandBar = Nothing

End Sub

Private Sub testFunktionCommandBarButton_Click(ByVal ctrl As _
    Office.Core.CommandBarButton, ByRef cancelDefault As Boolean) _
    Handles testFunktionCommandBarButton.Click
    MsgBox(Testfunktion())
End Sub

Private Sub dataBaseInfoCommandBarButton_Click(ByVal ctrl As _
    Office.Core.CommandBarButton, ByRef cancelDefault As Boolean) _
    Handles dataBaseInfoCommandBarButton.Click
    MsgBox(accessApplication.CurrentDb.Name)
End Sub

End Class
```

Anm.: Einige Probleme hatte ich bei der Speicherverwaltung mit COM-Objekten. Der Gargabe Collector kümmert sich zwar um die .net-Objekte, bei den COM-Objekten führt die verzögerte Freigabe aber oft zu Problemen.

In diesem Fall hilft `GC.Collect()` bzw. `Marshal.ReleaseComObject(...)` oft weiter.

Eine weitere Funktion, die Abhilfe bringen kann ist `GC.WaitForPendingFinalizers()`.

Unter <http://access.joposol.com/download/BeispielComAddIn.zip> steht das Beispiel-Projekt zum Download bereit.