

Beispiel 3

Beispiel zur Erstellung eines COM-Add-Ins über Visual Studio.NET

In diesem Beispiel wird ein COM-Add-In erstellt, das Grafiken in PowerPoint Folien einfügt. Dabei wird für jede Grafik eine neue Folie hinzugefügt.

Neues Projekt erstellen

Nachdem Sie im Startfenster auf "Neues Projekt" geklickt haben, wählen Sie in der Liste der Projekttypen

Andere Projekte / Erweiterungsprojekte

Aus der Liste der Vorlagen entscheiden Sie sich für

Gemeinsames Add-In

Im unteren Bereich geben Sie den Dateinamen und den Speicherort des Add-Ins an. Klicken Sie auf den Button "Vergrößern", wenn Sie einen extra Ordner für die Dateien anlegen möchten.

Der Add-In Assistent

Es öffnet sich ein Assistent, der Sie durch die Grundeinstellungen bringt:

Wählen Sie die Programmiersprache:

Visual Basic

Wählen Sie einen Anwendungshost:

Microsoft PowerPoint

Geben Sie einen Namen und eine Beschreibung ein:

Name: PowerPointAddIn

Beschreibung: Einfügen von Grafiken in PP-Folien

Wählen Sie die Add-In Optionen

Aktivieren Sie beide Optionen, damit das Add-In beim Start der Anwendung geladen wird und es auch für alle User zur Verfügung steht.

Zusammenfassung

Hier werden Ihnen alle Punkte nochmals angezeigt, bevor der Assistent die Entwicklungsumgebung vorbereitet.

Der Assistent bereit nun die Umgebung nach Ihren Einstellungen vor.

Im Projektmappen-Explorer sind die grundsätzlichen Objekte angelegt:

Verweise

Die notwendigen Verweise sind bereits vom Assistenten eingetragen worden. Meistens benötigt man jedoch auch den Verweis auf die Hostanwendung, diese können Sie selbst nachtragen. Klicken Sie den Ordner "Verweise" mit der rechten Maustaste an und fügen die benötigten Verweise hinzu. In diesem Beispiel: Microsoft PowerPoint

Connect.cs

Hier finden Sie die wichtigen Ereignisse der `IDTExtensibility2`, die als Schnittstelle zur Hostanwendung fungiert. Klicken Sie mit der rechten Maustaste auf dieses Objekt und wählen aus dem Kontextmenü "Code anzeigen".

Der Assistent hat den Code bereits für Sie vorbereitet. Aus Gründen der Übersichtlichkeit stelle ich vor und nach dem eigenen Code einen auskommentierten Hinweis.

Im Deklarationsbereich finden Sie die Zeilen:

```
Imports Microsoft.Office.Core
imports Extensibility
imports System.Runtime.InteropServices
```

Folgende Namespaces fügen Sie hier hinzu:

```
' *****Beginn eigener Code *****
Imports Microsoft.Office.Core.MsoControlType
Imports Microsoft.Office.Core.MsoButtonStyle
Imports Microsoft.Office.Core.MsoFileDialogType
Imports Microsoft.Office.Core.MsoFileDialogView
Imports Microsoft.Office.Core.MsoTriState
Imports PowerPoint.PpSlideLayout
' *****Ende eigener Code *****
```

Jede Datei kann beliebig viele Imports-Anweisungen enthalten. Imports-Anweisungen müssen vor Deklarationen, einschließlich Module- oder Class-Anweisungen, und vor Verweisen auf Identifizierer platziert werden.

Als erste Zeile in der Class-Anweisung finden Sie die Einbindung der IDTExtensibility2:

```
Implements Extensibility.IDTExtensibility2
```

Damit werden die notwendigen Ereignisse

```
OnConnection
OnDisconnection
OnStartupComplete
OnBeginShutdown
OnAddInsUpdate
```

bereit gestellt. Nähere Information, wann welches Ereignis aufgerufen wird, entnehmen Sie bitte dem Script "AEK5_Addin".

Es werden alle 5 Ereignisse im Class-Block eingetragen, diese dürfen auch nicht entfernt werden. Es ist aber nicht notwendig in jedem Ereignis Code einzutragen.

Außerdem sind hier 2 Variablen deklariert

```
Dim applicationObject As Object
Dim addInInstance As Object
```

Fügen Sie eine weitere Variable für den Menüleisteneintrag in der Hostanwendung ein:

```
' *****Beginn eigener Code *****
Dim WithEvents objCommandBarButton As CommandBarButton
' *****Ende eigener Code *****
```

Im Ereignis "**OnStartupComplete**" schreiben Sie den Code, der in der Hostanwendung den Menüleisteneintrag erstellt:

```
' *****Beginn eigener Code *****
Dim objCommandBars As CommandBars
Dim objCommandBar As CommandBar
Dim objCommandBarControl As CommandBarControl

'Erstellen eines Menübefehls unter "Extras"
objCommandBars = applicationObject.commandBars
objCommandBar = objCommandBars.Item("Tools")

'Prüfen ob dieser Eintrag schon existiert
For Each objCommandBarControl In objCommandBar.Controls
    If objCommandBarControl.Caption = "Grafik einfügen..." Then
        objCommandBar.Controls.Item("Grafik einfügen...").Delete()
    End If

Next objCommandBarControl
objCommandBarButton=objCommandBar.Controls.Add(msoControlButton)

With objCommandBarButton
    .Caption = "Neue Grafik einfügen..."
    .Style = msoButtonCaption
    .Tag = "Grafik einfügen..."
    .OnAction = "!<PowerPointAddIn.Connect>"
    .Visible = True
End With
' *****Ende eigener Code *****
```

Im Ereignis "**OnDisconnection**" wird auf das Schliessen der Hostanwendung reagiert:

```
' *****Beginn eigener Code *****
On Error Resume Next
' Die Verbindung zum Add-In kann unterbrochen werden, da die Anwendung
geschlossen wurde
applicationObject = Nothing
' *****Ende eigener Code *****
```

Im Ereignis "**OnConnection**" wird auf das Ereignis OnStartupCompleet verwiesen:

```
' *****Beginn eigener Code *****
'Unabhängig wie die Anwendung gestartet wird, das Add-In starten
If (connectMode <> Extensibility.ext_ConnectMode.ext_cm_Startup) _
Then Call OnStartupComplete(custom)
' *****Ende eigener Code *****
```

Damit sind die Ereignisse abgearbeitet, es fehlt nur noch der Code, der bei Klick auf den Menüleisteneintrag aufgerufen werden soll.

```
' *****Beginn eigener Code *****
Private Sub objCommandBarButton_Click(ByVal Ctrl As CommandBarButton,
ByRef Canceldefault As Boolean) Handles objCommandBarButton.Click

'erstellen eines OpenFileDialog, um die vom User gewählten Dateien
auszuwählen
Dim objOpenFileDialog As OpenFileDialog = _
applicationObject.FileDialog(msoFileDialogOpen)
Dim objSelectedItem As Object
Dim objPresentation As PowerPoint.Presentation
With objOpenFileDialog
.AllowMultiSelect = True
.Title = "Select Graphics Files"

'Nur bitmap, GIF und JPGs zulassen
.Filters.Clear()
.Filters.Add("Grafikdatei auswählen", "*.bmp; *.gif;*.jpg")
.InitialView = msoFileDialogViewThumbnail

' -1 = wenn der User öffnen klickt
If .Show = -1 Then
objPresentation = applicationObject.presentations.add
For Each objSelectedItem In .SelectedItems

'eine Folie pro selectierter Datei einfügen
objPresentation.Slides.Add(1, ppLayoutBlank)_
.Shapes.AddPicture(objSelectedItem, msoFalse, msoTrue, 0, 0)
Next objSelectedItem
End If
End With
End Sub
' *****Ende eigener Code *****
```

Damit ist die Erstellung des Add-Ins fast abgeschlossen.

Gehen Sie jetzt über "Debuggen" auf "Starten" und testen Ihr Add-In.

Ist dieser Test erfolgreich abgeschlossen, können Sie die Projektmappe, die DLL und auch das Setup erstellen:

Rufen Sie über das Menü "Erstellen" den Eintrag "PowerPointAddin erstellen" auf. Damit werden alle benötigten Dateien in den am Anfang bestimmten Ordner eingefügt.

Im Ausgabefenster sollte am Ende der Hinweis erfolgen, dass die Erstellung erfolgreich abgeschlossen wurde.