

ASP und COM: Austausch ohne Neustart

Die Pflege von ASP-Anwendungen macht es notwendig, Komponenten auszutauschen. Dabei gilt es, Neustarts des Webservers zu vermeiden und die Erreichbarkeit der Seiten so weit wie möglich zu erhalten. Dieser Beitrag zeigt, wie das möglich ist.

___ Wenn Code aus den Visual-Basic-COM-Komponenten in ASP-Seiten oder gar in Stored Procedures ausgelagert wird, erhöht das deren Entropie. Der Grund ist häufig, dass die Funktionalität der Komponente angepasst werden muss und neue Versionen auf dem Produktivserver eingespielt wurden. Das bedeutet üblicherweise einen Neustart zumindest des IIS, oft aber auch des Servers.

Als einer der großen Vorteile von ASP.NET wird hervorgehoben, dass .NET-Komponenten im laufenden Serverbetrieb ausgetauscht werden können. Der Resonanz auf diese Ankündigung zufolge scheinen viele Entwickler mit diesem Problem zu kämpfen.

Es besteht darin, dass eine COM-DLL vom IIS-Prozess (*Inet-Info.exe*) geladen und im Speicher gehalten wird. Solange aber eine Datei im Speicher aktiv ist, kann sie nicht auf der Festplatte ausgetauscht werden. Eventuell ist es möglich, den w3svc-Dienst, respektive den iisadmin-Dienst durch Eingabe von *net stop iisadmin* in der Kommandozeile, die DLL auszutauschen und dann mit *net start w3svc* den w3svc-Dienst wieder zu starten. Allerdings sind dann keine Webs erreichbar, die auf diesem Server laufen. Diese Methode ist nicht immer erfolgreich und beeinflusst unter Umständen mehrere Anwendungen, da der Server neu gebootet werden muss.

Klassisches ASP ist durch die Vermischung von Programmcode mit HTML-Elementen und einer Reihe weiterer Nachteile, die sich durch VB-Script und die Code-Organisation unter ASP ergeben, sehr unübersichtlich. Der IIS ist in der Lage, ASP-Anwendungen in jeweils eigenen Prozessen auszuführen. Dieser eine

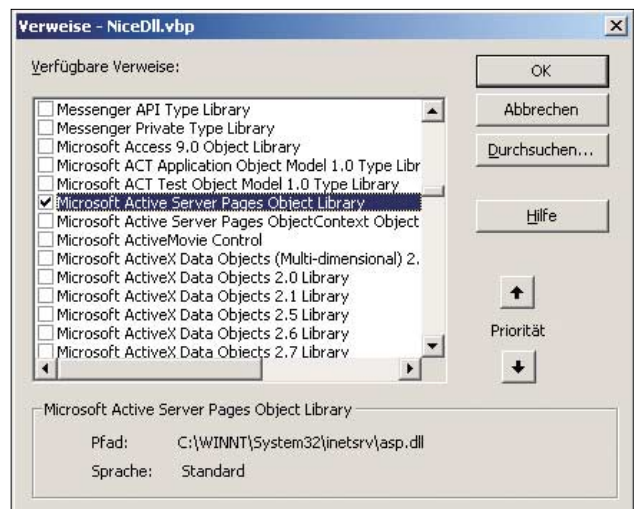


Abbildung 1 | Eine Referenz auf die ASP-Objektbibliothek.

Prozess, also eine Webanwendung, nicht der ganze Webserver, muss dann beendet werden. Danach ist die DLL austauschbar.

Eine COM-Komponente für ASP

Zunächst wird eine COM-Komponente in Visual Basic entwickelt, die nicht einfach nur ein Funktionsergebnis zurückgibt, sondern direkt in die ASP-Seite hineinschreibt. Die Vorteile liegen auf der Hand: Die Komponente kann in einer Sprache Ihrer Wahl geschrieben werden, besteht aus kompiliertem Code, ist typisiert und kann leicht mit einer Visual-Basic-Anwendung getestet werden.

SUMMARY

Auf einen Blick

COM-Komponenten einer klassischen ASP-Anwendung lassen sich ohne Neustart des Servers oder des Webdienstes austauschen.

Eingesetzte Anwendungen

Visual Basic 5 oder 6, IIS 5, ASP

Autor

Marcel Gnoth ist Senior Consultant bei der Berliner NTeam GmbH. Seine Arbeitsschwerpunkte liegen im Bereich COM, .NET und verteilte Informationssysteme, für die er auch Trainings durchführt. Sie erreichen ihn unter marcel@gnoth.net oder www.gnoth.net.

L 1 Der Quellcode der VB-Klasse.

```
Option Explicit
Private m_ScriptingContext As ScriptingContext
Public Sub OnStartPage(sc As ScriptingContext)
    Set m_ScriptingContext = sc
End Sub
Public Sub Addiere(a As Long, b As Long)
    m_ScriptingContext.Response.Write "<HR>"
    m_ScriptingContext.Response.Write "Das Ergebnis der Addition von: _
    " & a & " + " & b & " ist: "
    m_ScriptingContext.Response.Write "<B>" & a + b & "</B>"
    m_ScriptingContext.Response.Write "<HR>"
End Sub
```

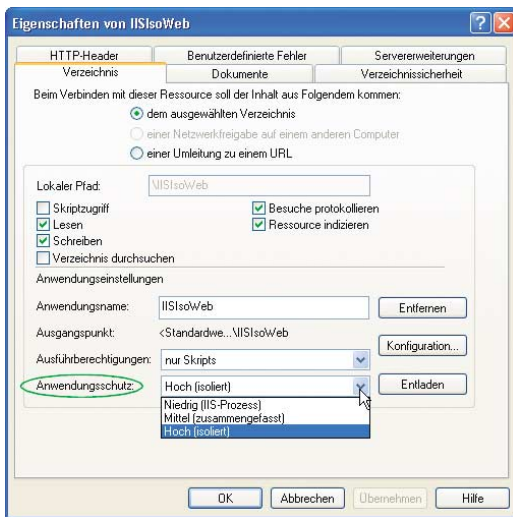


Abbildung 2 | Der Anwendungsschutz legt fest, in welchem Prozess die Anwendung und damit auch die DLL ausgeführt wird.

Öffnen Sie ein neues VB-DLL-Projekt, benennen Sie das Projekt *NiceDll* und das Klassenmodul *HappyASPClass* und fügen Sie eine Referenz (Menü *Projekt – Verweise*) auf die *Microsoft Active Server Pages Object Library* hinzu (vergleiche *Abbildung 1*). Dadurch können Sie in Ihrem VB-Projekt die ASP-Objekte *Response*, *Request* und so fort benutzen. Um von der VB-DLL in die ASP-Seite zu schreiben und die ASP-Objekte benutzen zu können, ist eine Referenz der ASP-Seite notwendig, welche die Komponente aufruft. Die Referenz muss zur Laufzeit erfolgen. Es gilt folgende Aufrufkonvention: Wenn Sie *Server.CreateObject* in einer ASP-Seite aufrufen und eine COM-Komponente instanzieren, versucht der IIS die Methode *OnStartPage* der Komponente aufzurufen. Ist sie implementiert, erhalten Sie eine Referenz auf die ASP-Seite (*Listing 1*).

Mit dieser Referenz können zur Laufzeit alle ASP-Objekte benutzt werden, genauso wie durch Skript-Code, der sich direkt auf der ASP-Seite befindet. Das ist die Konstruktionsweise von Webklassen, denen ein Recordset und einige Parameter übergeben werden. Sie schreiben dann zur Laufzeit den kompletten HTML-Code für eine Tabelle oder ein Listenfeld in die ASP-Seite. So wird die Ausführung aufgrund des kompilierten Codes schneller und der Code der ASP-Seite viel übersichtlicher.

Komponente installieren und aufrufen

Nun wird ein neues Web im IIS angelegt, was mit Visual Interdev sehr einfach geht. Das Web heißt *IISIsoWeb*, die hinzugefüg-

te ASP-Seite *Default.asp*. Die ASP-Seite enthält Code, der die Komponente instanziiert (siehe *Listing 2*).

Um das Verhalten der *OnStartPage*-Methode zu untersuchen, wird VB 6 gestartet und es werden Haltepunkte in den beiden Methoden der VB-6-COM-Komponente gesetzt. Wird die ASP-Seite im Browser aufgerufen, hält VB 6 zweimal an. Zunächst bei *CreateObject*, wenn der IIS der Komponente die Referenz auf das ASP-Objekt übergibt, und dann beim Aufruf der Methode *Addiere*. Arbeiten Sie mit Visual Interdev, können Sie nahtlos eine ASP-Seite und die VB-Komponente debuggen. Allerdings ist das Einrichten des Debuggens unter Visual Interdev nicht ganz trivial. Informationen dazu finden sich in der MSDN.

Nachdem VB 6 gestoppt wurde, wird die Komponente kompiliert und die ASP-Seite erneut aufgerufen. Da der *InProcServer32*-Key in der Registry für die Komponente auf die kompilierte DLL gesetzt ist, wird die DLL in den Prozessraum des Aufrufers geladen. Möchten Sie jetzt die DLL neu kompilieren oder mit einer neuen Version überschreiben, fangen die Probleme an.

Die folgenden Ausführungen beziehen sich auf IIS 5 unter Windows 2000 und Windows XP. In [1] und [2] finden Sie die Informationen für NT 4 und IIS 4. Für eine Webanwendung unter IIS 5.0 gibt es drei Ausführungsmöglichkeiten, die über die Eigenschaften eines Webs konfiguriert werden (*Computerverwaltung – Internet-Informationdienste – Standardwebseite – Eigenschaften Ihres Webs*, siehe *Abbildung 2*). Stellen Sie den Anwendungsschutz auf

L 2 Der Code der ASP-Seite zum Testen der Komponente.

```
<%@ Language=VBScript %>
<%Option Explicit%>
<HTML>
<HEAD>
<Title>NiceDll</Title>
</HEAD>
<BODY>
<h1>Testen der ASP-Komponente</h1>
<%
Dim o
Set o = Server.CreateObject("NiceDll.HappyASPClass")
Call o.Addiere(25,35)
%>
</BODY>
</HTML>
```

Niedrig (IIS-Prozess), werden das Web und die DLL im Prozess des Internet Information Servers (*InetInfo.exe*) ausgeführt. Der Vorteil ist die schnellere Interprozesskommunikation zwischen Ihrem Web und dem Webserver. Verursacht das Web oder die Komponente einen bösen Absturz, kann der gesamte Webdienst mit in den Abgrund gerissen werden. Möchten Sie die DLL austauschen, müssen Sie den Dienst neu starten. Das betrifft alle Webs. Als Alternative dazu kann die Anwendung in einem separaten Prozess *dllhost.exe* ausgeführt werden. Steht der Anwendungsschutz auf *Hoch (isoliert)*, wird für die Anwendung nur ein einziger eigener *dllhost*-Prozess gestartet, die Einstellung *Mittel (zusammengefasst)* erlaubt Ihnen mehrere Webanwendungen. Durch die getrennten Prozessräume bleiben die einzelnen Anwendungen und der IIS voreinander geschützt. Dieser Vorteil wird mit Performanceeinbußen durch die Interprozesskommunikation erkaufte. Dafür können Sie jetzt Webs, die in einem eigenen Prozess laufen, herunterfahren, ohne den Server oder andere Webs zu beeinflussen.

Austausch der DLL

Stellen Sie bei den Eigenschaften Ihres Webs den Anwendungsschutz auf *Hoch* und starten Sie anschließend die Management-Console für die Komponentendienste COM+ (*Startmenü – Verwaltung – Komponentendienste*). Hier sehen Sie die COM+-Anwendungen (*Abbildung 3*).

Unter dem Knoten COM+-Anwendungen befindet sich ein Eintrag für die Webanwendung (IISIsoWeb). Dieser Eintrag verschwindet, wenn Sie den Anwendungsschutz wieder auf *Niedrig* umstellen und die Ansicht aktualisieren. Wird die Anwendung im Browser aufgerufen, startet ein neuer *dllhost*-Prozess, in dem die ASP-Anwendung und die VB-Komponente ausgeführt werden. In der Komponentendienste-Verwaltung ist unter dem Knoten *Ausgeführte Prozesse* der Prozess des Webs zu sehen. Dieser kann mit dem Kontextmenü heruntergefahren werden, wodurch er und damit auch die DLL aus dem Speicher entfernt werden. Die DLL kann überschrieben oder neu kompiliert werden, ohne andere Webs zu beeinflussen. Beim nächsten Aufruf des Webs im Browser wird der *dllhost*-Prozess automatisch neu gestartet.

Wird der Anwendungsschutz bei den Eigenschaften des Webs umgestellt, müssen Sie nicht über die Konsole der Komponentendienste gehen. Stellen Sie einfach temporär den Anwendungsschutz auf *Niedrig* und der *dllhost*-Prozess verschwindet, anschließend kompilieren Sie neu und stellen den Anwendungsschutz wieder auf *Hoch*.

Flexibler Anwendungsschutz

Die Performance und die Sicherheit bei Abstürzen müssen eine hohe Priorität erhalten. Beim Austausch der Komponenten ist mehr Flexibilität nötig. In der Praxis ist das nicht schwer. Während der anfänglichen Entwicklungs- und Testphase laufen die Webs in eigenen Prozessen. Später, wenn das Web in den Routinebetrieb übergeht und maximale Performance notwendig ist, wird der Anwendungsschutz auf *Niedrig* umgeschaltet. Funktionalität in COM-Komponenten auszulagern macht Ihre Webanwendung übersichtlicher, schneller und wartungsärmer. |||||

- [1] Server Reliability through Process Isolation, J.D.Meier, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnserv/html/server112299.asp>
- [2] Ted Pattison, COM- und MTS-Programmierung mit Visual Basic, Microsoft Press

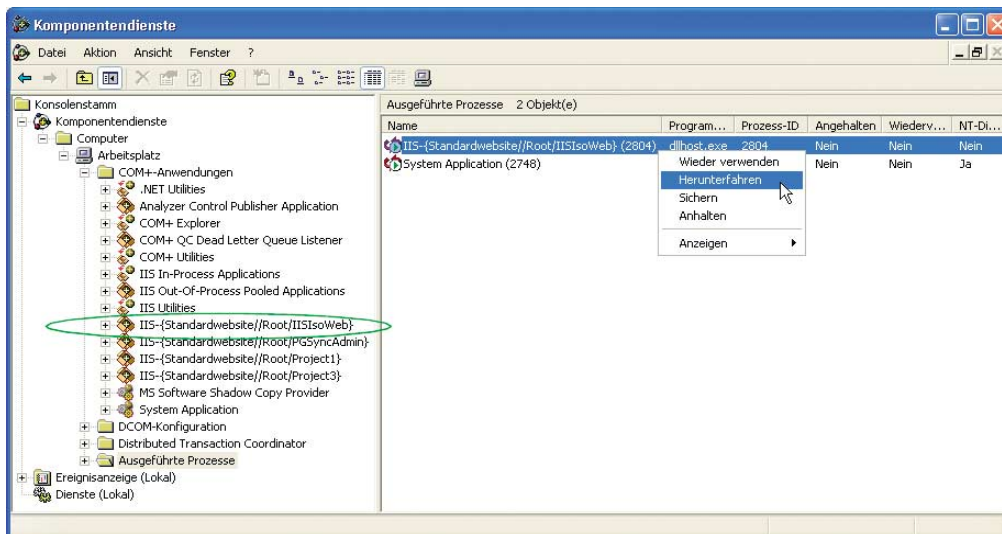


Abbildung 3 | Mit der Verwaltung der Komponentendienste COM+ können Sie gezielt einzelne Prozesse stoppen.