

Whitepaper

Produkt: combit address manager & combit Relationship Manager Debuggen von Scripts in Visual Studio 2005

Inhalt

Script in Visual Studio 2005 debuggen	3
Einleitung	3
Schritt für Schritt zum Debuggen von Scripts	3
Debuggen von internen Scripts in Visual Studio 2005	10
Voraussetzungen	10
Schritt für Schritt zum Debuggen von internen Scripts	10

Script in Visual Studio 2005 debuggen

Einleitung

Der Debugger in Visual Studio war bisher sehr gut, um z.B. Windows Anwendungen zu debuggen. Man kann so schnell einen Fehler finden oder sich einfach Ausgaben anzeigen lassen. Dasselbe funktioniert auch mit VBScripts oder JScripts. Auf den nächsten Seiten ist erklärt, wie das Ganze funktioniert.

Schritt für Schritt zum Debuggen von Scripts

Zuerst wird ein neues Projekt über File > New > Project... angelegt. Als nächstes wird ein Makefile Project angelegt wie in Abbildung 1 gezeigt.

New Project	2 🔀
Project types:	Iemplates:
Visual C++ ATL CLR General MFC Smart Device Win32 Other Languages Other Project Types	Visual Studio installed templates Custom Wizard Makefile Project My Templates Search Online Templates
A project for using an external b	uild system
Name: VbsProject	
Location: C:\Dokumen	te und Einstellungen\MarkoP\Eigene Dateien\Visual Studio 2005\Projects 🛛 🛛 📴 📴 🔤 rowse
Solution Name: VbsProject	Create directory for solution
	Add to Source Control
	OK Cancel

Abbildung 1

Nachdem der OK – Button geklickt wurde erscheint der Application Wizard. Hier einfach auf "Finish" klicken.

- 3 -

Danach werden im Solution Explorer (siehe Abbildung 2) drei Ordner und eine Textdatei angelegt. Diese werden nicht benötigt und können somit gelöscht werden.

Solution Explorer - Solution 'YbsProject' (1 project) 🔯
🧓 Solution 'VbsProject' (1 project)
🚊 🚰 ¥bsProject
🔤 Header Files
🔤 Resource Files
🚞 Source Files
🗐 readme.txt
Solution Explorer 🐼 Class View 📑 Property Manager
Abbildung 2

Anschließend klickt man rechts auf das Projekt und fügt ein bestehendes oder neues Script ein (siehe Abbildung 3). In diesem Fall nehme ich ein bestehendes VBScript. Dieses Script sollte anschließend in den Projektordner kopiert werden.

	Build	1		
	Rebuild	I		
	Clean	I		
	Project Only	I		
	Custom Build Rules	I		
	Tool Build Order	I		
	Add •	Ē		New Item
	References	E		Existing Item
	Add Web Reference	Ē	*	New Filter
	Set as StartUp Project	4	23	Class
	Debug +	Ę	23	Resource
2	Add Solution to Source Control	I		
Ж	Cut	I		
	Paste	I		
×	Remove	I		
	Rename			
	Unload Project			
	Properties			

Abbildung 3

Daraufhin klickt man rechts auf das Projekt und wählt den Menüpunkt "Properties" aus (siehe Abbildung 4) .

📥 🚧 VheDr	. aim wsno.	ner
👘 👘 🖏 📖	Build	≥1
	Rebuild	7 A F
	Clean	16
	Project Only	
	Custom Build Rules	
	Tool Build Order	
	Add 🕨	1
	References	
	Add Web Reference	
	Set as StartUp Project	
	Debug 🕨	
2	Add Solution to Source Control	
*	Cut	
12	Paste	
×	Remove	
	Rename	
	Unload Project	
	Properties	

Abbildung 4

Hier wird nun der "Configuration Type" von "Makefile" in "Utility" geändert (siehe Abbildung 5). Danach auf den Punkt "Debugging" klicken.

VbsProject Property Pages			? 🔀
Configuration: Active(Debug)	Platform: Active(Win32)	· · · · · · · · · · · · · · · · · · ·	Configuration Manager
 Common Properties Configuration Properties General Debugging NMake 	Output Directory Intermediate Directory Configuration Type Build Log File Build Log File Configuration Type Specifies the type of output this content of the type of output this content of the type of output typ	figuration generates.	
		ОК	Abbrechen Übernehmen

Abbildung 5

Hier erscheint dann folgendes Fenster (siehe Abbildung 6). In der Command-Zeile trägt man folgenden Pfad ein:

> C:\WINDOWS\system32\wscript.exe

Und in der Command Arguments-Zeile wird folgendes eingetragen: > //X Scriptname (in diesem Fall WshShell.vbs)

Das "//X" bewirkt, dass das VBScript im Debug-Modus ausgeführt wird.

VbsProject Property Pages			? 🗙
Configuration: Active(Debug)	Platform: Active(Win32)	Configuration	Manager
Common Properties Eferences Configuration Properties	Debugger to launch: Local Windows Debugger		~
General Debugging ⊕- Build Events	Command Command Arguments Working Directory Attach Debugger Type Environment Merge Environment SQL Debugging	C:\WINDOWS\system32\wscript.exe //X WshShell.vbs No Auto Yes No	
	Command The debug command to execute.		
		OK Abbrechen Ü	<u>b</u> ernehmen

Abbildung 6

Anschließend einfach einen Breakpoint im Quellcode setzen (siehe Abbildung 7).

WshShell.vbs
Option Explicit
dim WshShell, objEnv
Set WshShell = WScript.CreateObject("WScript.Shell")
Set objEnv = WshShell.Environment("System")
MsgBox objEnv("OS")

Abbildung 7

Danach kann das Script ausgeführt werden. Jedoch nicht mit "F5", sondern mit "Strg + F5". Dies bewirkt, dass kein neues Projekt für dieses Script angelegt wird. Es kann nun im selben Fenster gedebuggt werden. Sobald man die Tastenkombination "Strg + F5" gedrückt hat, erscheint folgendes Fenster (siehe Abbildung 8):

Visual Studio Just-In-Time Debugger 🛛 🔀
An unhandled exception ('Script Breakpoint') occurred in wscript.exe [3088].
Possible Debuggers:
New instance of Visual InterDev 6.0/Visual J++ 6.0 New instance of Visual Studio 2005
VbsProject - Microsoft Visual Studio: Visual Studio 2005
Set the currently selected debugger as the <u>d</u> efault.
Manually choose the debugging engines.
Do you want to debug using the selected debugger?
Yes No

Abbildung 8

Hier wählt man folgenden Debugger aus:

 \rightarrow Projektname – Microsoft Visual Studio: Visual Studio 2005

Jetzt kann man das Script ganz normal debuggen (siehe Abbildung 9).

W	/shShell.vbs 📾
	Option explicit
•	<pre>dim WshShell, objEnv Set WshShell = WScript.CreateObject("WScript.Shell") Set objEnv = WshShell.Environment("System") MsgBox objEnv("OS")</pre>

Abbildung 9

Wenn man jetzt "Strg + Leertaste" drückt, unterstützt Visual Studio 2005 sogar IntelliSense für die Basis Script-Funktionen (siehe Abbildung 10).



Abbildung 10

Debuggen von internen Scripts in Visual Studio 2005

Voraussetzungen

Interne Scripte lassen sich nur mit dem combit Relationship Manager ab Version 2007.009 oder dem address manager 14 debuggen. Zusätzlich benötigt man natürlich auch hier Visual Studio 2005 oder einen anderen Windows Debugger.

Schritt für Schritt zum Debuggen von internen Scripts

Als erstes müssen im Script folgende zwei Zeilen eingefügt werden (siehe Abbildung 11).

Besipiel anhand eines VBScripts:

```
14
       Option Explicit
15
16
       <!--#pragma debugmode-->
17
      ' Aktives cRM Projekt
18
19
       Dim oProject, oListViews, oView
20
       Set oProject = cRM.CurrentProject
       Set oListViews = oProject.ActiveViews
21
       Set oView = oListViews.ActiveView
22
23
      Wscript.DebugBreak
24
25
       MsgBox "Aktive Ansicht: " & oView.Name, vbInformation, "combit Relationship Manager"
26
27
28
       ' .... <eigener Scriptcode> ....
29
30
      ' Objekte freigeben:
      Set oView = Nothing
31
32
      Set oListViews = Nothing
33
      Set oProject = Nothing
34
```

Abbildung 11

Die Zeile "<!--#pragma debugmode-->" aktiviert den internen Debugmodus. Die zweite Zeile "Wscript.DebugBreak" verhält sich wie ein Breakpoint und aktiviert den Debugger.

Wenn das Script jetzt intern gestartet wird, z.B. durch einen Button-Klick, erscheint das Fenster in dem man den Debugger auswählen kann (siehe Abbildung 12).

Visual Studio Just-In-Time Debugger 🛛 🔀
An unhandled exception ('Script Breakpoint') occurred in wscript.exe [3088].
Possible Debuggers: New instance of Visual InterDev 6.0/Visual J++ 6.0
New instance of Visual Studio 2005
Set the currently selected debugger as the <u>d</u> efault.
<u>Manually choose the debugging engines.</u>
Do you want to debug using the selected debugger?
<u>Y</u> es <u>N</u> o

Abbildung 12

Hier wählt man folgenden Debugger aus: → New instance of Visual Studio 2005

Jetzt lässt sich das Script zur Laufzeit in Visual Studio 2005 debuggen.

Hinweis: combit macht keine Angaben zu einer bestimmten Eignung obiger Informationen. Irrtümer und Fehler bleiben ausdrücklich vorbehalten, die Angaben erfolgen ohne Gewähr und enthalten keine Zusicherung. Die Informationen können z.T. auch ein Versuch sein, Ihnen bei einer Aufgabenstellung zu helfen, selbst wenn das Produkt eigentlich nicht für diesen speziellen Zweck vorgesehen wurde.