1 DOTNET Scripting Host (DSH) 1.1

Autor: Holger@Schwichtenberg.de
Stand: 06.01.2003
DSH-Version: 1.1.3

DSH

Der DOTNET Scripting Host (DSH) ist ein Active Scripting Host für das .NET Framework, der die wesentlichen Funktionen des ActiveX Scripting-basierten Windows Script Host (WSH) für die Common Language Runtime nachbildet:

- Start von Skripten an der Kommandozeile
- Verpacken von Skripten in einer XML-Datenstruktur
- Entfernte Ausführung von Skripten (Fernausführung, engl. Remoting)

Darüber hinaus bietet der DSH gegenüber dem WSH auch einige erweiterte Funktionen wie die Protokollierung in eine Textdatei oder ein Windows-Ereignisprotokoll.

Voraussetzungen

Der DSH funktioniert bei installierten .NET Framework 1.0 oder 1.1 Beta1. Eine Kompatibilität zu früheren oder späteren Versionen ist nicht ausgeschlossen, aber nicht getestet.

Hintergrundkom pilierung

Der DSH arbeitet nicht mit Script for .NET/Visual Studio for Applications (VSA), sondern verwendet die in der .NET Framework Class Library (FCL) integrierten Compiler direkt. Der DSH nutzt diese Compiler, um eine übergebene Quellcodedatei zu kompilieren und dann die erzeugte Assembly auszuführen. Die Assembly wird dabei im Speicher erzeugt und (im Standard) nicht auf die Festplatte geschrieben. Der Nutzer des DSH kann daher nicht erkennen, dass das Programm nicht interpretiert, sondern kompiliert wird. Derzeit unterstützt der DSH keine anderen .NET-fähigen Sprachen außer Visual Basic .NET, C# und JScript .NET. Die optionale Speicherung der erzeugten Assembly ist möglich.

Freeware

Der DSH ist Freeware und steht derzeit als Version 1.1 zur Verfügung. Autor des DSH ist Holger Schwichtenberg (hs@IT-Visions.de). Der DSH ist zu finden im Internet unter http://www.windows-scripting.de.

Installation

Der DSH ist implemtiert in DSH.EXE. Kopieren Sie diese Datei auf Ihr System. Wenn Sie möchten, dass die Dateiextension .DSH mit der dsh.exe verknüpft wird, tragen Sie bitte in der mitgelieferten Datei installdsh.reg den Pfad zur DSH.EXE auf Ihrem System ein und starten Sie dann diese Datei, um die notwendigen Einträge in der Registry vorzunehmen.

Sprachversione

Alle Ausgaben des DSH erfolgen auf Deutsch, wenn die auf dem System eingestellte Sprache Deutsch ist. In allen anderen Fällen erfolgt die Ausgabe in Englisch.

1.1 Dateiformat

Der DSH verwendet als Dateiformat XML. Das ist sinnvoll, weil neben dem eigentlichen Script auch noch Zusatzinformationen wie der Name der verwendeten Sprachen und der referenzierten Assemblies an den Compiler übergeben werden müssen.

Ein Skript-Dokument kann mehrere einzelne Skripte enthalten, die sequentiell ausgeführt werden.

XML-Element	Erlaubte Häufigkeit	Erläuterung
`<?xml version="1.1" encoding="ISO-8859-2"?>`	1	Processing Instruction
`<scriptdoc>`	1	Wurzelelement, das den Rahmen um alle untergeordneten Elemente bildet. Mit dem Attribut `process="1"` wird festgelegt, dass eventuell schon vorher laufende Instanzen dieses Skriptdokuments gewaltsam beendet werden.
`<comment>`	0..N	Kommentar-Element, wird vom DSH ignoriert.
`<references>`	0/1	Fasst ein oder mehrere `<assembly>`-Elemente zusammen.
`<assembly>`	0..N	Definiert eine zu referenzierende Assembly.
`<log>`	0/1	Legt fest, welche Meldungen des DSH in ein Ereignisprotokoll und/oder eine Textdatei geschrieben werden sollen.

Attribut	Erläuterung
`error`	Optionales Attribut Sofern das Attribut existiert und nicht auf `"NO"` steht, wird ein Protokolleintrag im Fehlerfall erzeugt.
`success`	Optionales Attribut Sofern das Attribut existiert und

<table>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>nicht auf <code>"NO"</code> steht, wird ein Protokolleintrag bei erfolgreicher Beendigung eines Skripts erzeugt.</td></tr>
<tr><td><code>start</code></td><td>Optionales Attribut

Sofern das Attribut existiert und nicht auf <code>"NO"</code> steht, wird ein Protokolleintrag beim Start eines Skripts erzeugt.</td></tr>
</table>
</td></tr>
<tr><td><code><eventlog></code></td><td>0/1</td><td>Unterelement von <code><log></code>.

Legt im Attribut <code>name</code> den Namen des Ereignisprotokolls fest, in das die Einträge geschrieben werden sollen. Außer den Standardprotokollen (z.B. Application, System und Security) können auch beliebige eigene Protokollnamen verwendet werden. Der DSH legt automatisch ein entsprechendes Ereignisprotokoll an.</td></tr>
<tr><td><code><logfile></code></td><td>0/1</td><td>Unterelement von <code><log></code>.

Legt im Attribut <code>name</code> den Pfad einer Textdatei fest, in die die Protokolleinträge geschrieben werden sollen.</td></tr>
<tr><td><code><script></code></td><td>1..N</td><td>Quellcode des Skripts. Es kann beliebig viele <code><script></code>-Elemente geben. Die Skripte werden nacheinander in der Reihenfolge ihres Auftretens in der Datei abgearbeitet.
<table>
<tr><th>Attribut</th><th>Erläuterung</th></tr>
<tr><td><code>Name</code></td><td>Optionales Attribut

Name für das Skript</td></tr>
<tr><td><code>language</code></td><td>Notwendiges Attribut

Mögliche Attributwerte: <code>"VB.NET"</code>, <code>"CSharp"</code> und <code>"JS.NET"</code>.</td></tr>
<tr><td><code>startClass</code></td><td>Optionales Attribut

Name der Klasse, deren Unterroutine <code>Main()</code> den Startpunkt des Skripts bildet. (Im Standard wird nach einer Klasse mit Namen <code>Main</code> gesucht, also <code>Main::Main()</code>.)</td></tr>
</table>
</td></tr>
</table>

Tabelle 1.1: XML-Elemente in DSH-Skripten

Die Dateiextension ist beliebig. Vorgesehen ist aber die Dateiextension .DSH.

Beispiel

Beispiel

Das folgende Listing zeigt ein Beispiel mit drei Skripten:

? Das erste Skript (CSharp) gibt Hello World aus.

? Das zweite Skript (VB.NET) liest die Zeichenkette "Hello World" aus einer XML-Datei aus.

? Das dritte Skript (VB.NET) listet die übergebenen Parameter auf.

```
<?xml version="1.1" encoding="ISO-8859-2"?>
<scriptdoc process="1">

<comment>
Demo-Skript von Holger Schwichtenberg
</comment>

<references>
<assembly>system.xml.dll</assembly>
<assembly>System.DirectoryServices.dll</assembly>
<assembly>system.data.dll</assembly>
</references>

<log error="YES" success="YES" start="YES">
<eventlog1 name="Scripting"/>
<logfile1 name="e:\log.txt"/>
</log>

<!-- ************************************** -->

<script name="FirstScript" language="CS"
startClass="Scripting.StartKlasse">
using System;
namespace Scripting
{
       class StartKlasse
       {
```

```
            static void Main(string[] args)
            {
                  Console.WriteLine("FirstSkript: Hello World!");
            }
      }
}
</script>

<!-- ************************************** -->

<script name="SecondScript" language="vb.net">
<![CDATA[
Imports System.XML
Imports System

Module Main
    Sub Main()

        ' --- Test XML
       Dim d as new XmlDocument
        d.loadxml("<test><message>Hello World!</message></test>")
       Console.Writeline("SecondSkript: Message from the document: " &
d.SelectSingleNode("*//message").InnerText)


     End Sub
End Module
]]>
</script>

<!-- ************************************** -->

<script name="ThirdScript" language="vb.net">
<![CDATA[
```

```
Imports System

Module Main
    Sub Main(args as string())
    Console.WriteLine("ThirdScript: List auf script arguments:")
         ' --- Argumente ausgeben
            Dim s as string
            for each s in args
                  console.writeline("- "& s)
          next
    End Sub
End Module
]]>
</script>
</scriptdoc>
```

Listing 1.1: Beispiel für ein DSH-Skript
[CD:/code/DOTNET_Scripting/DSH/Beispiel1.dsh]

1.2 Aufbau des Skripts

Main()

Ein Skript muss einen eindeutigen Einsprungpunkt besitzen. Dazu muss ein Skript aus mindestens einer Klasse (ein Modul in Visual Basic ist eine spezielle Form einer Klasse) mit einer statischen Methode bestehen, die `Main()` heißt. Optional sind weitere Attribute und Methoden in dieser Klasse oder weitere Klassen mit Attributen und Methoden möglich.

startClass

Der Name der Klasse, die den Einsprungpunkt bildet, ist frei wählbar (definierbar durch das XML-Attribut `startClass` im `<script>`-Element). Der Name der Methode ist nicht frei wählbar, er lautet immer `Main()`. Es gibt zwei mögliche Signaturen für `Main()`:

? `Sub Main()`

? `Sub Main(args as string())`

Parameter

Im letzteren Fall empfängt `Main()` vom DSH die dem Skript übergebenen Kommandozeilenparameter in einem `Array of String`. Andere Parameter sind bei `Main()` nicht erlaubt. Der Name, der dem Parameter gegeben wird, ist frei wählbar.

```
Module Main
    Sub Main(Argumente as string())
```

```
    ' …
    End Sub
End Module
```

Listing 1.2: Grundgerüst für ein DSH-Skript

1.3 Start eines Skripts

Man startet ein DSH-Skript, indem man DSH.EXE mit dem Dateinamen aufruft.

dsh.exe

```
H:\DSH>dsh.exe testscript.dsh
---------------------------------
DOTNET Scripting Host (DSH)
Version: 1.1
(C) Holger Schwichtenberg 2002
http://www.windows-scripting.com
---------------------------------

FirstSkript: Hello World!
SecondSkript: Message from the document: Hello World!
ThirdScript: List auf script arguments:
- ../demos/testscript.dsh
```

1.4 Kommandozeilenparameter

Ebenso wie der WSH unterstützt der DSH drei Arten von Kommandozeilenparametern:

Kommandozeilenparameter

? Name (inklusive Pfad) des aufrufenden Skripts. Der erste Parameter, der nicht mit einem Slash (/) beginnt, wird als Dateiname betrachtet.

? Parameter für den DSH selbst. Diese beginnen mit einem doppelten Slash (//).

? Parameter für das aufgerufene Skript. Alle Parameter, die nicht mit einem doppelten Slash (//) beginnen, werden dem aufgerufenen Skript als Parameter übergeben. Das Skript erhält also auch seinen eigenen Dateinamen. Dieser wird immer als erster Parameter an das Skript übergeben.

Parameter	Erläuterung
//ABOUT	Liefert Informationen über den DSH.

`//HELP`	Zeigt Hilfe zu den Kommandozeilenoptionen an.
`//EN`	Alle Ausgaben in englischer Sprache.
`//DE`	Alle Ausgaben in deutscher Sprache.
`//S`	Silent. Der DSH macht – wenn kein Fehler auftritt – keine Ausgaben außer denen, die das Skript selbst erzeugt.
`//V`	Verbose. Der DSH ist sehr "geschwätzig" und liefert zusätzliche Ausgaben über die Verarbeitung der XML-Datei und den Übersetzungsvorgang.
`//ERRORGUI`	Legt fest, dass Fehlermeldungen des DSH als Dialogboxen angezeigt werden sollen.
`//SUCCESSGUI`	Legt fest, dass nach erfolgreicher Ausführung aller Skripte eine Dialogbox angezeigt werden soll.
`//OUT:Verzeichnis`	Legt fest, dass die kompilierten Assemblies im Dateisystem in dem angegebenen Verzeichnis gespeichert werden sollen.
`//SERVER:Portnummer`	Startet den DSH als Server (Listener). Der DSH wartet auf dem angegebenen Port auf (Fern-)Aufrufe von einer anderen Instanz des DSH. Der DSH-Listener ist so lange aktiv, bis der Prozess beendet wird.
`//COMPUTER:Name`	Name oder IP-Adresse des Computers, auf dem das Skript gestartet werden soll. Ohne diese Angabe wird das Skript auf dem lokalen Computer gestartet.
`//PORT:Portnummer`	Portnummer der entfernten DSH-Instanz. Ohne diese Angabe wird das Skript auf dem lokalen Computer gestartet.
`//HTTP`	Als Protokoll für das Remote Scripting wird HTTP anstelle von TCP verwendet. Diese Option kann sowohl in Kombination mit `//COMPUTER` als auch `//SERVER` verwendet werden. Auch im Fall der Verwendung von `//HTTP` muss zusätzlich `//PORT` angegeben werden.

Tabelle 1.2: Kommandozeilenparameter des DSH 1.1

1.5 Eingebaute Objekte

Intrinsic Objects, FCL

Der DSH besitzt keine speziellen eingebauten Objekte (Intrinsic Objects). Den Skripten stehen alle Klassen der .NET Framework Class Library (FCL) zur Verfügung – genauso wie einem "normalen" .NET-Programm auch.

1.6 Assembly-Referenzen

Um eine Klasse zu nutzen, muss die Assembly, die diese Klasse implementiert, als Referenz hinzugefügt werden. Dazu muss im Element `<assembly>` der Name der Datei, die die Assembly enthält, angegeben werden. In dem Element darf nicht auch der Pfad genannt sein.

Für den Standort der referenzierten Assembly gilt es zwei Fälle zu unterscheiden:

? Im Normalfall muss die Datei muss im gleichen Verzeichnis wie die *dsh.exe*, nicht wie das Skript, liegen. Eine Installation im GAC ist leider derzeit nicht möglich, weil die Compiler dort nicht nach einer Assembly suchen können.

? Eine Ausnahme bilden Assemblies aus der FCL: Da es von allen diesen Assemblies eine Kopie im .NET Framework-Verzeichnis gibt, werden die Assemblies gefunden. (Diesen Trick könnte man auch auf selbstgeschriebene Assemblies anwenden, dann muss man dieser aber an zwei Stellen pflegen!)

Es ist sinnvoll, aber nicht notwendig, die zu verwendenden Namespaces mit `Imports` oder `using` einzubinden.

1.7 Assembly-Persistenz

Speicherung de erzeugten Assembly

Mit der Option `//OUT:Verzeichnis` kann die erzeugte Assembly im Dateisystem persistent gemacht werden, also das Skript in einer kompilierten Form abgelegt werden. Dieses Kompilat ist MSIL-Code, der direkt gestartet werden kann.

Die Assembly erhält dabei den Namen des Skripts innerhalb des Skriptdokuments. Als Dateiextension wird `.EXE` angehängt. Nach `//OUT:` muss das Verzeichnis angegeben werden, in dem die Assembly abgelegt werden soll. Wenn dort schon eine gleichnamige Datei existiert, wird diese ohne Warnung überschrieben. Das angegebene Verzeichnis muss existieren, sonst kommt es zu einer Fehlermeldung.

Wenn das Skriptdokument mehrere Skripte enthält, werden mehrere einzelne Assemblies erzeugt.

1.8 Fernausführung von Skripten

Remoting

Der DSH unterstützt Remoting. Man kann den DSH auf einem Computer aufrufen und ihn anweisen, ein lokal erreichbares Skript auf einem entfernten Computer auszuführen. Dies entspricht dem Konzept des Remoting beim Windows Script Host (WSH). Anders als beim WSH ist jedoch auf dem Client selbst kein Skript notwendig: Das Remoting wird durch eine Kommandozeilenoption des DSH initiiert.

Auf dem Server muss eine Instanz des DSH laufen, die zuvor auf einem bestimmten Port gestartet wurde. Auch dies ist anders als beim WSH, entspricht jedoch dem Remoting-Konzept des .NET Frameworks.

TCP/IP

Die Kommunikation zwischen Client und Server erfolgt in DSH über TCP/IP über einen frei wählbaren Port.

Start des Servers

Der Server wird gestartet durch die Option `//server` unter Angabe einer freien Portnummer. Wenn der Port noch frei ist, beginnt die Überwachung des Ports auf eingehende Aufrufe.

```
H:\DSH>dsh.exe //server:9999
DOTNET Scripting Host (DSH)
Version: 1.1
(C) Holger Schwichtenberg 2002
http://www.windows-scripting.com
Starting DSH listener on port 9999...
Press enter to stop this process.
```

Wenn der Port schon belegt ist, erscheint folgende Fehlermeldung:

```
ERROR: Normalerweise darf jede Socketadresse (Protokoll, Netzwerkadresse
oder Anschluss) nur jeweils einmal verwendet werden.
```

Das "Normalerweise" in diesem Satz wirkt fehlplatziert. Dies ist jedoch eine in der .NET Framework Class Library hinterlegte Meldung.

Start des Client

Der Client wird ganz normal unter Angabe eines Skriptnamens gestartet. Zusätzlich ist Folgendes festzulegen: der Computername (oder die IP-Adresse) nach `//computer` und der Port (`//port`), auf dem der entfernte DSH "lauscht".

```
dsh.exe testscript.dsh //computer:byfang //port:9999 //v
```

So sieht ein erfolgreicher Aufruf aus:

```
DOTNET Scripting Host (DSH)
Version: 1.1
(C) Holger Schwichtenberg 2002
http://www.windows-scripting.com
```

Im "geschwätzigen" Modus (`//V`) gibt es mehr Infos:

```
DOTNET Scripting Host (DSH)
Version: 1.1
(C) Holger Schwichtenberg 2002
http://www.windows-scripting.com
Loading Script Document...testscript.dsh
Connecting to Client byfang:9999...
Calling Remote DSH...
Script Document testscript.dsh finished successfully!
```

Der Client erhält die Fehlermeldung, wenn die Ausführung nicht erfolgreich war:

```
DOTNET Scripting Host (DSH)
Version: 1.1
(C) Holger Schwichtenberg 2002
http://www.windows-scripting.com
Loading Script Document...testscript.dsh
Connecting to Client byfang:9999...
Calling Remote DSH...
COMPILE ERROR in Script at Line # 10: BC30451:Name 'irgendwo1' is not
declared.
COMPILE ERROR in Script at Line # 15: BC30451:Name 'xyz' is not declared.
```

Ausgaben auf dem Server

Bitte beachten Sie, dass alle Bildschirmausgaben des übermittelten Skripts auf dem Server im DOS-Fenster des Serverprozesses stattfinden!

```
DOTNET Scripting Host (DSH)
Version: 1.1
(C) Holger Schwichtenberg 2002
http://www.windows-scripting.com
Starting DSH listener on port 9999...
Press enter to stop this process.
FirstSkript: Hello World!
SecondSkript: Message from the document: Hello World!
ThirdSkript: List auf script arguments:
- testscript.dsh
- /abc
- testoption
```

Wenn der Server mit der Option `//v` gestartet wird, informiert er sehr ausführlich:

```
DOTNET Scripting Host (DSH)
Version: 1.1
(C) Holger Schwichtenberg 2002
http://www.windows-scripting.com
Starting DSH listener on port 9999...
Press enter to stop this process.
```

```
Receiving call...
Parsing document...
Parsing References...
Adding References: system.xml.dll
Adding References: System.DirectoryServices.dll
Adding References: system.data.dll
Parsing Script Document...
usw.
```

1.9 Limitationen

Fehlende Features

Derzeit beim DSH (noch) nicht möglich ist,

? dass sich Skripte innerhalb einer Skriptdatei gegenseitig aufrufen.

? dass Skripte andere Skriptdokumente referenzieren. Referenzen sind nur auf Assemblies möglich.