Helma Spona

Einstieg ins Windows Scripting

Galileo Computing

Liebe Leserin, lieber Leser,

Sie haben sich für ein Galileo Computing-Buch zum Thema »Windows Script Host« entschieden und liegen damit voll im Trend: Programmierung ist out, Scripting ist in. Der Windows Script Host ermöglicht die Ausführung von Visual Basic und JavaScript. Das Buch zeigt Voraussetzungen, Gefahren und Nutzen des WSH auf und führt in die Programmierung mit VBScript und dem WSH-Objektmodell ein. Verwendet wird der WSH 5.6, der in Windows XP integriert ist, aber für andere Windows-Versionen ebenso installiert werden kann.

Eine Besonderheit des Buches ist seine genaue Leserführung. Unsere erfahrene Autorin Helma Spona beweist in diesem Buch erneut, dass sie auszuwählen versteht zwischen wesentlichen und marginalen Inhalten. Dass sie erkennt, welche Inhalte den zukünftigen Skriptprogrammierer interessieren und mögliche Fehlerquellen gleich mit ihm zusammen umschifft. Sie lernen, ohne dass es an einer Stelle mühsam und praxisfern wird.

Doch kein noch so gutes Computerbuch ist ohne Fehler. Deshalb freue ich mich auf Ihre kritische Mitarbeit. Zögern Sie nicht, mich direkt anzuschreiben, um mir Ihre Verbesserungsvorschläge mitzuteilen.

Und nun viel Freude beim Lesen!

Judith Stevens-Lemoine
Lektorat Galileo Computing

judith.stevens@galileo-press.de
www.galileocomputing.de

Galileo Press • Gartenstraße 24 • 53229 Bonn

Auf einen Blick

Inhalt

Vorwort

Der WSH ist die ideale Entwicklungsumgebung, nicht nur um kleinere Programmieraufgaben zu bewältigen und wiederkehrende Vorgänge in Windows zu automatisieren, sondern er eignet sich auch hervorragend, um programmieren zu lernen. Sie brauchen nämlich nicht erst Software im Wert von ein paar hundert Euro, sondern bekommen alles, was Sie brauchen, vom Editor bis zum Debugger und der Laufzeitumgebung kostenlos, die passende Windows-Version vorausgesetzt.

Mit diesem Buch werden Sie nicht nur langsam in die Programmierung eingeführt, Sie können auch schon nach den ersten Kapiteln nützliche kleine Programme erstellen, die Ihnen das Leben mit Windows leichter machen. Am Ende beherrschen Sie zudem die wichtigen Programmierkonzepte, die es Ihnen ermöglichen, Ihre Kenntnisse um andere Programmiersprachen wie VBA und Visual Basic zu erweitern und neben Windows auch die Office-Anwendungen, CorelDRAW oder andere VBA-Hostanwendungen zu steuern.

Und natürlich erfahren Sie in diesem Buch auch, wie Sie Windows trotz WSH sicher machen. Mit den passenden Sicherheitseinstellungen und einem aktuellen Virenscanner ist das gar kein Problem. Der WSH und eine sichere Windows-Oberfläche widersprechen sich also keinesfalls.

Die Beispiele zum Buch finden Sie auf der Website des Verlags: **www.galileo-computing.de**. Entpacken Sie die ZIP-Datei bitte mit den Verzeichnissen, da die Skripte nach Kapiteln sortiert sind und in einigen Verzeichnissen Skripte mit gleichen Namen vorkommen.

Sollten Sie Fragen haben oder Kritik anbringen wollen, können Sie mich über das Kontaktformular auf meiner Website erreichen: **www.helma-spona.de**. Dort finden Sie auch Fehlerberichtigungen.

Und nun viel Spaß beim Programmieren mit dem WSH und gutes Gelingen.

Ihre Helma Spona

1 Einführung

1 Einführung

Nach Melissa und Co. ist der Windows Script Host ganz zu Unrecht von vielen Administratoren als das Übel schlechthin angesehen worden. Jedoch nicht der WSH ist das Problem, sondern der Benutzer, der vor dem Rechner sitzt.

Der Windows Script Host (WSH) hat in den letzten Jahren vor allem durch die immer wieder auftauchenden Skript-Viren von sich reden gemacht. Trotzdem ist und bleibt sein Nutzen hoch, wenn man die Gefahren kennt, korrekt einschätzt und erst denkt und dann klickt. Wo die Gefahren und Möglichkeiten des WSH liegen, wie Sie ihn installieren und welche Programme es zur Quelltexterfassung gibt, zeigt Ihnen dieses erste Kapitel.

1.1 Was kann der WSH?

Ursprünglich war der Windows Script Host als Nachfolger der Batch-Programmierung von MS DOS geplant. Deren Möglichkeiten übertraf aber schon die Version 1 des Windows Script Host, die 1998 veröffentlicht wurde. Damals hieß er noch Windows Scripting Host. Schon diese erste Version war in der Lage, die Registry zu manipulieren und auf andere Anwendungen, wie etwa die Microsoft-Office-Programme, zuzugreifen und das Dateisystem zu manipulieren. Für die Batch-Verarbeitung war der WSH 1.0 daher ausgesprochen gut geeignet. Mittlerweile ist die Version 5.6 des WSH aktuell. Damit ist es sogar möglich, Skripte auf anderen Rechnern im Netzwerk zu starten (Remote Scripting), Skripte zu signieren und die Skriptausführung unsignierter Skripte mit Sicherheitseinstellungen zu unterbinden oder einzuschränken. Vor allem die Sicherheit ist das wichtigste Argument zum Einsatz der neuesten Version.

Skripte für den WSH lassen sich ganz einfach mit jedem Texteditor erstellen und ausführen. Sie brauchen also weder eine teure Entwicklungsumgebung, noch müssen Sie Ihre Hardware aufrüsten, um die Skripte benutzen zu können. Jeder kann Skripte erstellen und schreiben. Aus diesem Grund eignen sie sich hervorragend, um wiederkehrende Alltagsaufgaben zu lösen wie:

- Festplatte aufräumen, Dateien komprimieren, umbenennen oder in Verzeichnisse aufteilen
- Startmenüeinträge erstellen
- kleine Setup-Programme erstellen
- Registry-Einträge anpassen oder REG-Dateien einlesen

- fremde Programme steuern
- Datenbankabfragen ausführen

Grafische Benutzeroberflächen kennt der WSH jedoch nicht. Allerdings gibt es die Möglichkeit, den Internet Explorer und HTML für diesen Zweck zu benutzen, so dass Sie richtige kleine Programme mit dem WSH schreiben können. Neben den Aufgaben, die sich schon bisher mit BAT-Dateien mehr oder weniger elegant lösen ließen, bietet der WSH somit noch vieles mehr. Sie können nämlich Komponenten erstellen, die Sie nicht nur in WSH-Skripten, sondern auch in VBA- und Visual-Basic-Anwendungen einsetzen können.

1.1.1 Die Gefahren des WSH

Eine Gefahrenquelle für die Daten des Rechners ist der WSH nur dann, wenn man sorglos mit seinem Rechner umgeht. Sofern Sie erst denken und dann klicken, kann es nicht passieren, dass Sie versehentlich ein fremdes Skript ausführen und sich so den Rechner mit Viren und Würmern verseuchen. Alle Skriptdateien werden ordnungsgemäß von Windows als Skripte mit entsprechenden Symbolen kenntlich gemacht. Daran können Sie schon vor der Ausführung erkennen, dass es sich um eine Skriptdatei handelt. Als Symbol wird eine ausgerollte Skriptrolle angezeigt, die abhängig vom Dateityp unterschiedliche Farben haben kann. Aus dem Internet Explorer mit aktuellen Sicherheitsupdates kann eine Skriptdatei nur dann ausgeführt werden, wenn Sie extrem niedrige Sicherheitseinstellungen haben oder die Ausführung explizit bestätigen. Und hier gilt dann natürlich wieder: Erst denken, dann klicken.

Hinweis Gewappnet mit einem aktuellen Virenscanner passiert aber auch dann nichts, wenn Sie Skripte versehentlich ausführen. Heutige Virenscanner warnen rechtzeitig vor gefährlichen Befehlen in den Skripten und unterbinden standardmäßig deren Ausführung.

Vom WSH geht also mit minimalen Sicherheitsvorkehrungen keine größere Gefahr aus. Mehr zum Thema Sicherheit erfahren Sie in Kapitel 9, Sicherheit.

1.1.2 Was ist der WSH?

Der WSH selbst stellt lediglich zwei Programme, CScript und WScript, zur Verfügung, zwei Laufzeitumgebungen für Skripte. Sie sorgen dafür, dass der Skript-Quellcode eingelesen und auf Fehler geprüft wird und, falls er fehlerfrei ist, durch den Interpreter der jeweiligen Skriptsprache ausgeführt wird. Der WSH ist somit vergleichbar mit einer VBA-Hostanwendung. Auch sie stellt nur die Laufzeitumgebung für die VBA-Anwendungen dar.

Für alle (Noch-)Nicht-Programmierer: Die Laufzeitumgebung ist vergleichbar mit einer Eislaufbahn. Sie stellt den Läufern den Raum und die Technik zur Verfügung, die diese zum Schlittschuhlaufen benötigen. Laufen müssen sie jedoch selbst und können dies mit ganz verschiedenen Schlittschuhen tun.

Der WSH ermöglicht von sich aus die Programmierung in zwei Skriptsprachen, nämlich JScript und VBScript. Darüber hinaus kann er jedoch erweitert werden, so dass auch die Nutzung von PHP, Perl, Phyton, ReXX etc. denkbar ist. Der Interpreter muss dazu nur in den WSH integriert werden. Das ist ohne spezielle Schnittstellenprogrammierung des Interpreters aber nicht möglich. Er muss also speziell an den WSH angepasst werden.

Neben den Interpretern für JScript und VBScript sowie der Laufzeitumgebung stellt der WSH außerdem noch Objektbibliotheken zur Verfügung, mit deren Hilfe Sie erst die Möglichkeiten des WSH voll ausschöpfen können. Sie sind die Basis der WSH-Programmierung und können von allen Skriptsprachen des WSH genutzt werden.

1.1.3 Die Wahl der Skriptsprache

Welche Skriptsprache Sie nutzen möchten, hängt von Ihren persönlichen Vorlieben ab. Sowohl mit JScript als auch mit VBScript können Sie alle prinzipiell mit dem WSH lösbaren Aufgaben bewältigen. In einigen Fällen ist der Aufwand zwar unterschiedlich hoch, aber auf Dauer gleicht es sich wieder aus.

VBScript ist allerdings von der Syntax her einfacher zu lernen, vor allem, wenn Sie vielleicht schon Basic-, Visual-Basic- oder VBA-Kenntnisse mitbringen. VBScript ist nämlich eine Teilmenge von VBA und das wiederum stellt eine Teilmenge des Sprachumfangs von Visual Basic dar. Die Syntax von VBScript, das heißt, wie Befehle aufgebaut sind, ist fast identisch mit der von VBA und Visual Basic und die VBScript-Befehle, die Sie für den WSH benötigen, sind auch in VBA und Visual Basic enthalten.

VBScript verzeiht zudem Fehler, die JScript nicht gestattet. Gerade für den Einstieg ist VBScript daher die bessere Skriptsprache. Sie wird daher auch für alle Beispiele dieses Buches verwendet.

1.2 Den WSH 5.6 installieren

Bevor Sie loslegen können, müssen Sie aber den WSH installieren. Die aktuelle Version 5.6 ist bereits in Windows XP integriert. Wenn Sie Windows XP verwenden, können Sie die nachfolgenden Abschnitte überspringen und mit der

Installation der Dokumentation fortfahren. Für alle übrigen Windows-Versionen ab Windows 98 ist ein Update erforderlich, wenn das nicht schon erfolgt ist. Für Windows 95 steht der WSH leider nur bis zur Version 2.0 zur Verfügung. Sie können damit zwar auch einen Großteil der Beispiele nachvollziehen, einige erfordern jedoch zwingend die Version 5.6.

1.2.1 Die aktuelle WSH-Version ermitteln

Damit Sie wissen, ob Sie vielleicht schon die aktuelle Version installiert haben, sollten Sie die aktuelle Version des WSH ermitteln. Das ist recht einfach. Sie müssen dazu nur das Programm **Cscript.exe** starten. Gehen Sie dazu folgendermaßen vor:

1. Starten Sie zunächst die DOS-Box beziehungsweise die Eingabeaufforderung von Windows.
2. Geben Sie nun `cscript.exe` ↵ ein. Die WSH-Version wird nun in der zweiten Zeile ausgegeben. Steht hier ein Wert kleiner als 5.6, sollten Sie ein Update durchführen.

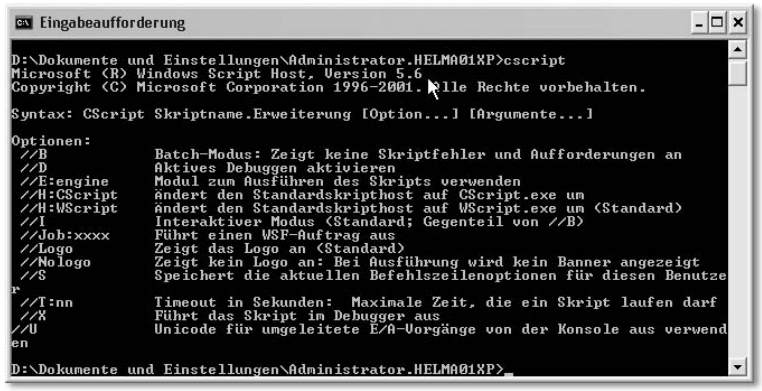

Abbildung 1.1 Nach Aufruf von cscript.exe wird in der zweiten Zeile die WSH-Version angezeigt.

1.2.2 Den WSH herunterladen

Die aktuelle Version des WSH finden Sie auf der Internetseite von Microsoft. Der Bereich, der sich ausschließlich mit Windows-Skripting beschäftigt, ist unter dem URL **www.microsoft.com/germany/scripting** zu finden. Leider ist es so, dass diese Seite extrem viele fehlerhafte Links enthält. Oftmals ist der WSH hierüber nicht herunterzuladen, weil der Link zur Downloadseite fehlerhaft ist. Versuchen können Sie es aber. Klicken Sie dazu einfach auf den Link **Windows Script 5.6 freigegeben**.

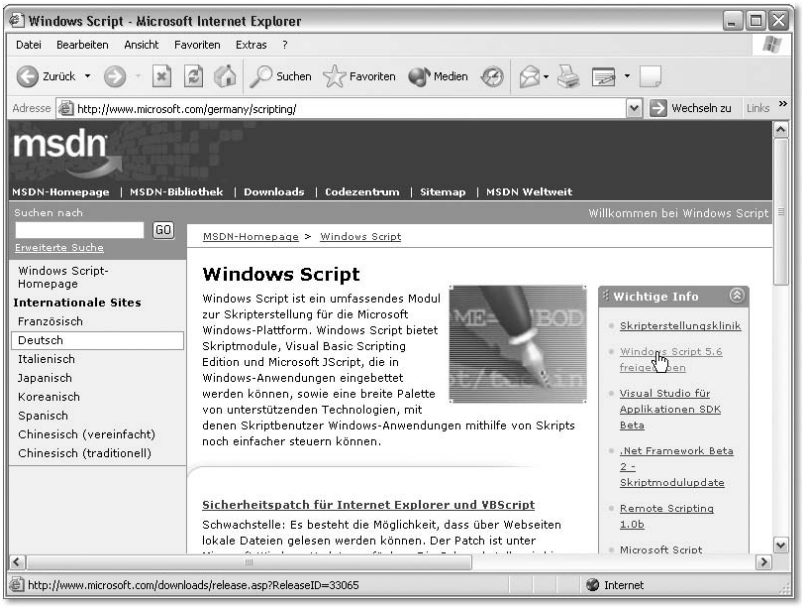

Abbildung 1.2 Windows Script 5.6 freigegeben

Wenn Sie Glück haben, gelangen Sie nun zur Download-Seite, auf der Sie nicht nur den WSH für Ihre Windows-Version herunterladen können, sondern auch die Dokumentation. Sollten Sie statt dessen die Meldung erhalten, dass die Seite nicht gefunden wurde, müssen Sie das Ganze anders angehen. Zuerst können Sie versuchen, ob die Seite unter folgendem URL erreichbar ist:

www.microsoft.com/downloads/details.aspx?displaylang=de&FamilyID=E74494D3-C4E1-4E18-9C6C-0EA28C9A5D9D

Wenn auch das nicht klappt, müssen Sie sich auf die Suche machen, indem Sie die Microsoft Download-Seite aufrufen.

1. Rufen Sie Ihren Browser auf und geben Sie als URL **www.microsoft.com/downloads/search.aspx?displaylang=de** ein. Sie gelangen dann in das deutsche Downloadcenter.
2. Geben Sie nun in das Feld **Stichwörter** den Begriff WSH 5.6 ein und klicken Sie auf **Go**.
3. Das Ergebnis der Suche sollten zwei Links zum Download des WSH sein. Abhängig von Ihrer Windows-Version brauchen Sie nun nur noch den entsprechenden Link anzuklicken und auf der Folgeseite rechts oben auf **Download** zu klicken.

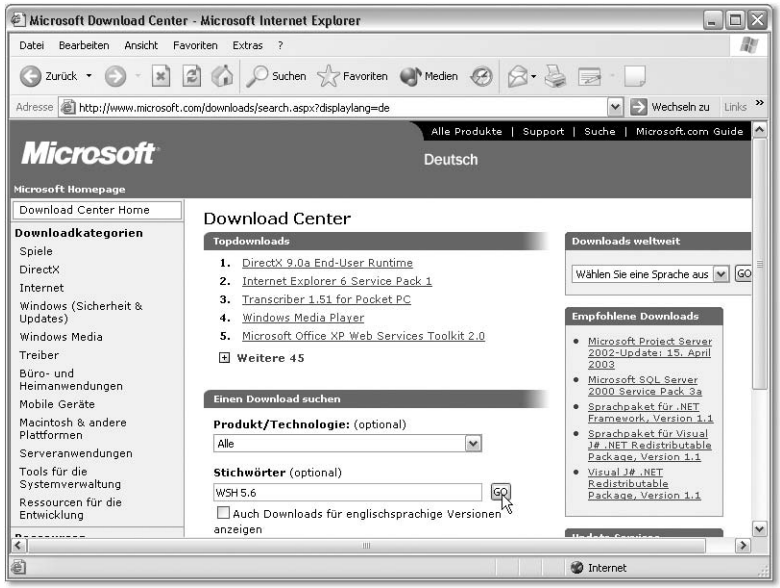

Abbildung 1.3 Suchen nach dem WSH-Download im Downloadcenter

1.2.3 Die Installation durchführen

Nach dem Download des WSH können Sie ihn installieren. Sie sollten dazu alle Office-2000-/XP-Anwendungen und auch alle Visual-Basic-6-Anwendungen schließen. Ansonsten kann es passieren, dass einige Dateien beim Setup nicht aktualisiert werden können. Falls Sie sich nicht sicher sind, beenden Sie einfach alle laufenden Programme.

> **Hinweis** Ganz wichtig ist, dass Sie in jedem Fall Ihren Virenscanner beenden oder vorübergehend deaktivieren. Die meisten Virenscanner verhindern nämlich die Installation des WSH. Vergessen Sie aber auf keinen Fall, den Virenscanner wieder zu aktivieren, wenn die Installation abgeschlossen ist.

Wenn Sie mit den Vorbereitungen fertig sind, brauchen Sie nur die Datei **scriptde.exe** (für Windows 2000 und XP) auszuführen beziehungsweise die Datei **scr56de.exe** für Windows 9x, Windows Me und Windows NT 4. Die Installation erfolgt dann vollkommen unproblematisch. Nach deren Abschluss sollten Sie allerdings den Rechner neu starten, damit die installierten Objektbibliotheken registriert werden und zur Verfügung stehen.

1.2.4 Die WSH-Dokumentation herunterladen

Für eine effektive Arbeit mit dem WSH ist die Dokumentation sinnvoll, die sogar in Deutsch verfügbar ist. Sie finden sie auf der regulären Download-Seite

des WSH, wenn es Ihnen gelingt sie über den URL **www.microsoft.com/germany/scripting** aufzurufen. Dann steht der Link zum Download ganz unten auf der Seite.

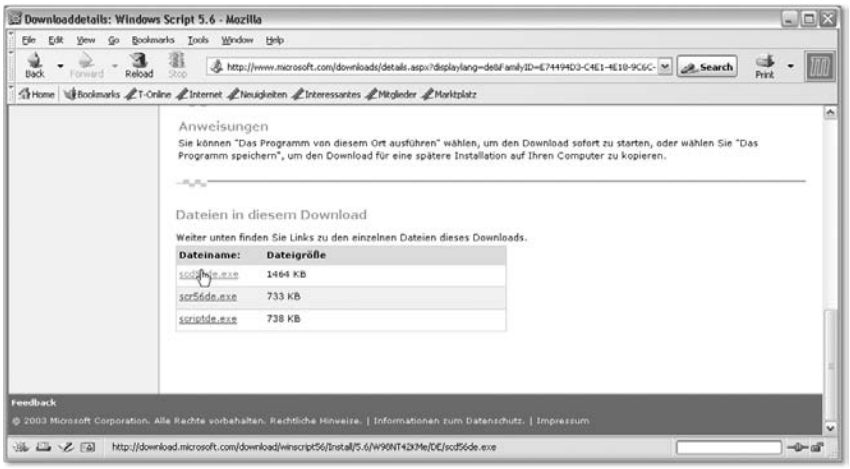

Abbildung 1.4 Der Link zum Download der Dokumentation

Wenn es Ihnen nicht gelingen sollte, diese Seite aufzurufen, versuchen Sie es mit folgendem URL:

http://download.microsoft.com/download/winscript56/Install/5.6/W98NT42KMe/DE/scd56de.exe

Falls auch das nicht gelingt, bleibt nur die Suche bei Google (**www.google.de**) übrig. Suchen Sie damit direkt nach der Datei **scd56de.exe**. Sie werden eine ganze Reihe Suchergebnisse erhalten, von denen sicherlich einige den passenden Download liefern.

1.2.5 Die Dokumentation installieren und starten

Nach dem Download lässt sich die Dokumentation ganz einfach per Doppelklick auf die EXE-Datei installieren. Im ersten Dialogfeld brauchen Sie nur den vorgeschlagenen Installationspfad zu bestätigen, können aber auch über **Durchsuchen** ein neues Verzeichnis auswählen. Klicken Sie dann auf **OK,** wird die Dokumentation ohne weitere Rückfragen installiert. Starten können Sie sie dann über **Start • Alle Programme • Microsoft Windows Script • Windows Script V5.6 Dokumentation**. Verwenden Sie nicht Windows XP, lautet der Eintrag entsprechend **Start • Programme • Microsoft Windows Script • Windows Script V5.6 Dokumentation**.

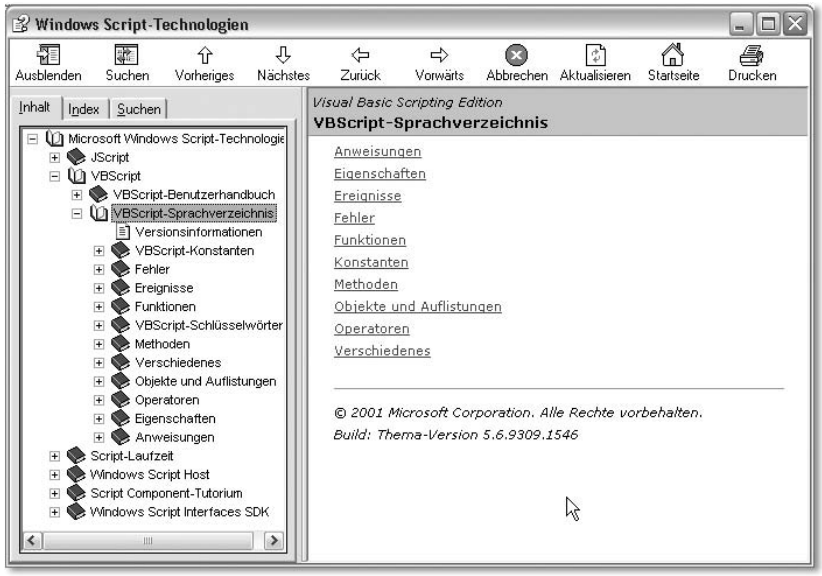

Abbildung 1.5 Die Dokumentation zum WSH 5.6

Im Inhaltsverzeichnis können Sie über den Eintrag **VBScript** direkt auf das VBScript-Handbuch und das Sprachverzeichnis zugreifen. Das sind die beiden wichtigsten Hilfen für den Anfang. Sie erläutern VBScript, ohne jedoch auf die Besonderheiten des WSH einzugehen. Mit VBScript können Sie nämlich nicht nur Skripte für den WSH schreiben, sondern auch Internetseiten interaktiv gestalten oder Outlook 97 und Outlook 98 programmieren. Details zum WSH finden Sie unter dem Punkt **Windows Script Host**.

1.3 Quellcode-Editoren

Neben dem WSH und der Dokumentation benötigen Sie zur Programmierung noch einen Editor, mit dem Sie den *Quellcode*, also den Programmtext der Skripte, erfassen können. Es eignet sich im Prinzip jeder Editor, der in der Lage ist, reine ASCII-Dateien ohne Formatierungen zu speichern. Im Zweifelsfall können Sie also den Windows-Editor verwenden. Der ist allerdings nicht sonderlich komfortabel, weil Sie zumindest in der in Windows 9x und Windows NT integrierten Version nicht einmal die Zeilen- und Spaltenanzahl an der Cursorposition angezeigt bekommen. Das ist aber für die Fehlersuche erforderlich. Für kurze Skripte ist der Windows-Editor daher zwar ausreichend, für längere und komplexere sollten Sie sich aber nach einer Alternative umsehen. Davon gibt es mittlerweile eine ganze Reihe, von denen die folgende Tabelle eine kleine Auswahl zeigt.

Editor	Autor/Hersteller	Download-Adresse	Beschreibung
Eddi	Helma Spona	www.helma-spona.de/buchdb/lstBuchdaten.php?ID=39	Ein einfacher Editor, der neben der Anzeige der Cursorposition auch die direkte Ausführung des Skriptes über den WSH ermöglicht.
Scripting Spy Trial	Dr. Tobias Weltner	www.scriptinternals.de	Der Editor bietet Syntaxhervorhebung und eine Programmierhilfe. Bei Eingabe bestimmter Schlüsselwörter werden Informationen zum Befehl angezeigt. Leider ist der Editor gelegentlich etwas träge.
NoteTab Light	Fookes Software	www.notetab.com/download.htm#ntl	Ein guter, einfacher Text- und HTML-Editor, der aber für WSH-Entwickler nichts Besonderes bietet.
VBScript Editor V.1.3	Koan Software	www.koansoftware.com/Eng/script.htm	Der Editor bietet eine farbige Hervorhebung von Quellcodebestandteilen und somit auch eine grobe Syntaxprüfung. Die direkte Ausführung der Skripte ist aber nicht möglich.
Primalscript	Sapien	www.primalscript.com/primalscript.htm	Während die vorstehenden Editoren alle kostenlos waren, ist Primalscript mit 179 US-$ nicht gerade preiswert. Es gibt aber zum Testen auch eine 30-Tage Testversion. Der Editor ist mit Sicherheit der beste, den es derzeit gibt. Er bietet nicht nur zahlreiche Programmierhilfen, sondern auch eine Projektverwaltung und Syntaxhervorhebung.

Tabelle 1.1 Überblick über nützliche Skript-Editoren

Neben den oben erwähnten Programmen können Sie aber auch jeden anderen Editor nutzen, der folgende Funktionen bietet:

- Speichern und Bearbeiten von Textdateien ohne Formatierung
- freie Wahl der Dateinamenserweiterung beziehungsweise Unterstützung der Erweiterung **.vbs**

- Anzeige der Zeilennummern oder Springen zu bestimmen Zeilen
- Suchen-Ersetzen

Weiterhin wären folgende Funktionen nützlich, aber nicht erforderlich:

- Starten des Skriptes direkt aus dem Editor
- Farbige Hervorhebung des Codes – *Syntaxhighlighting*
- Lesezeichen
- Programmierhilfen

Unter *Programmierhilfen* werden eine Reihe verschiedener Funktionen verstanden, die bei der Codeeingabe helfen. Dazu gehören die Vervollständigung von Schlüsselwörtern während der Eingabe (*Codevervollständigung*), die Anzeige von Konstanten und Elementen von Auflistungen (*Elementliste*) sowie die Anzeige von Informationen zu Aufrufparametern (*Parameterinfo*). Vor allem, wenn Sie in die Programmierung mit VBScript einsteigen und noch keinerlei Programmierkenntnisse in VBA und Visual Basic haben, werden Ihnen solche Programmierhilfen die ersten Schritte sehr erleichtern.

1.4 Das erste Skript erstellen und starten

Nach so vielen Vorbereitungen geht es nun an das erste Skript. Als Editor kommt hier die Trial-Version von Scripting Spy zum Einsatz. Wenn Sie einen anderen Editor verwenden, sollten Sie sich vorab mit den folgenden Grundfunktionen vertraut machen:

- neue Datei erzeugen
- speichern als Textdatei mit der Endung **.vbs**
- öffnen und bearbeiten vorhandener Dateien

1.4.1 Scripting Spy starten

Nach der Installation von Scripting Spy, die Sie einfach durch einen Klick auf die heruntergeladene MSI-Datei (MSI = Microsoft Installer) starten können, finden Sie im Startmenü von Windows einen Eintrag, über den Sie das Programm starten können: **Start • Alle Programme • Spy 3.0 Trial** beziehungsweise **Start • Spy 3.0 Trial**. Der erste Start kann unter Umständen etwas länger dauern, weil das Programm zunächst die für Skripte verfügbaren Objektbibliotheken des Systems analysieren muss. Haben Sie also etwas Geduld. Spätere Starts sind sehr viel schneller.

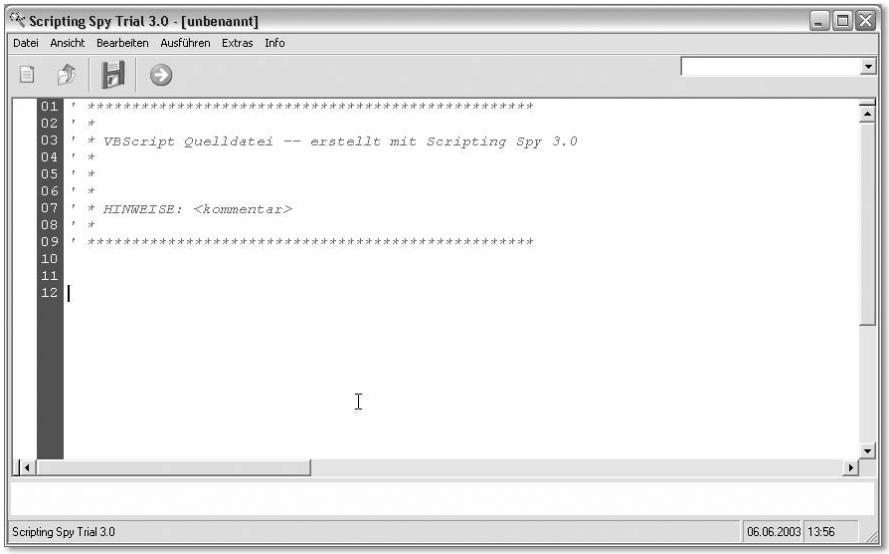

Abbildung 1.6 Scripting Spy nach dem Start

Scripting Spy erzeugt automatisch beim Start eine neue, leere Skriptdatei. Verwenden Sie einen anderen Editor, müssen Sie diese unter Umständen noch erzeugen. Erstellen Sie dazu eine einfache ASCII-Textdatei in Ihrem Editor.

1.4.2 Kommentarzeilen ergänzen

Neue Dateien in Scripting Spy beginnen mit neun Kommentarzeilen. Diese werden in Scripting Spy und vielen anderen Programmen mit Codehervorhebung grün dargestellt. Kommentarzeilen werden in VBScript durch ein Hochkomma eingeleitet. Alles, was danach bis zum Ende der Zeile folgt, ist ein Kommentar und wird bei der Ausführung des Skriptes nicht berücksichtigt. Sie können Kommentare also sehr gut zur Dokumentation Ihrer Skripte verwenden. Den Ansatz dazu zeigt Scripting Spy. Zeile sieben bietet bereits die Möglichkeit, den vorhandenen Platzhalter `<kommentar>` durch eine Beschreibung des Skriptes zu ersetzen.

1. Markieren Sie dazu zunächst den Platzhalter mit der Maus.
2. Überschreiben Sie ihn nun einfach mit Ihrem Wunsch-Kommentar, beispielsweise `Mein erstes Skript ...`

> **Hinweis** Was Sie in die Kommentarzeile einfügen, spielt keine Rolle. Beim Ausführen des Skriptes werden Kommentarzeilen behandelt, als wenn die Zeilen nicht vorhanden wären.

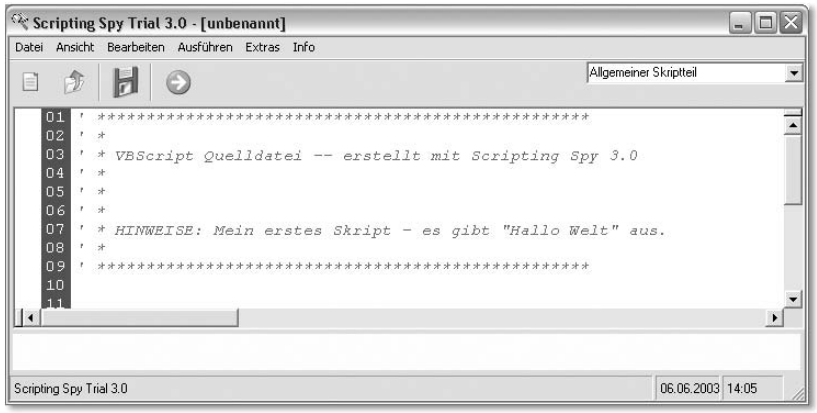

Abbildung 1.7 Der geänderte Kommentar

1.4.3 Die Programmierhilfen kennen lernen

Ausführbare Befehle können an jeder beliebigen Stelle in einem Skript stehen, also sowohl vor den Kommentarzeilen als auch danach oder zwischendrin. Nur innerhalb einer Kommentarzeile dürfen sie nicht stehen, weil sie dann auch als Kommentar behandelt werden.

Üblich ist es aber, Kommentare bezüglich der Aufgabe der Skriptes an den Anfang zu setzen und mit dem Skriptcode unterhalb dieser Kommentare zu beginnen. Wenn Sie also die erste Anweisung eingeben möchten, setzen Sie den Cursor einfach in Zeile zehn.

Das erste Beispiel soll eine Meldung in einem Dialogfenster ausgeben. Daran lassen sich dann auch sehr schön die Programmierhilfen von Scripting Spy demonstrieren. Sowohl der WSH als auch VBScript bieten verschiedene Möglichkeiten, eine Meldung auszugeben. Hier kommt die `MsgBox`-Funktion von VBScript zum Einsatz:

1. Beginnen Sie mit der Eingabe in Zeile zehn, indem Sie `msg` eingeben. Sofort öffnet sich ein kleines Fenster, das alle Befehle auflistet und gleich das Element markiert, das mit den eingegebenen Buchstaben beginnt. Hier ist es korrekt die gewünschte Funktion `MsgBox`.
2. Drücken Sie nun die ⇆-Taste, wird der Code vervollständigt und der markierte Befehl eingefügt. Diese Funktion wird *Codevervollständigung* genannt.
3. Im unteren, gelb unterlegten Teil des Fensters zeigt Scripting Spy nun die Parameterinformationen an. *Parameter* sind Werte, die an einen Befehl übergeben werden. Bei der `MsgBox`-Funktion definieren sie Aussehen und Inhalt des Textes, der als Meldung angezeigt wird. Die Parameter werden in Klam-

mern nach dem Befehlsnamen, hier `MsgBox`, angezeigt und durch Kommata voneinander getrennt. Das Wort `ByVal` gibt an, dass der Parameter als Wert übergeben werden muss.

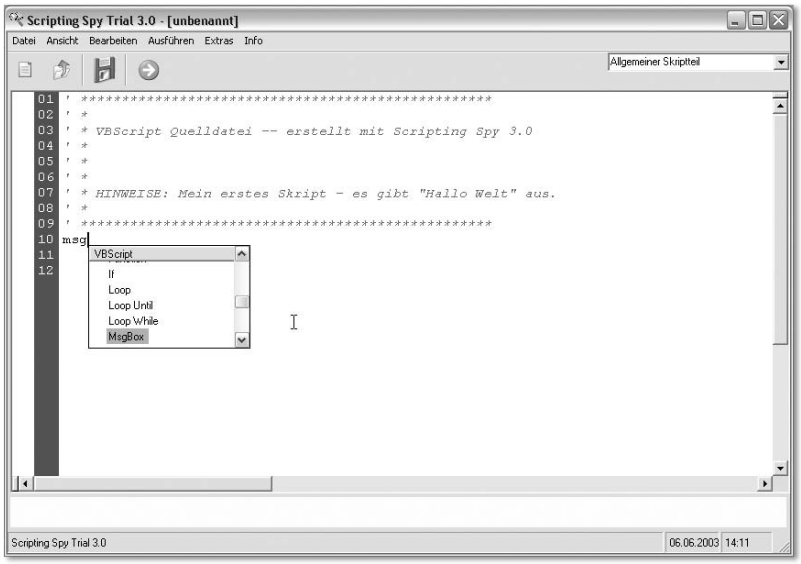

Abbildung 1.8 Die Elementliste von Scripting Spy

> **Hinweis** Welche Alternativen es sonst noch gibt und was dies im Detail bedeutet, erfahren Sie in Kapitel 2, Sprachgrundlagen von VBScript.

4. Danach wird immer der Name des Parameters angegeben. Steht er in Klammern, beispielsweise [buttons], muss er nicht angegeben werden, sondern ist optional. Danach folgt `As Variant`. Damit wird der Datentyp des Parameters angegeben. Da es in VBScript aber nur einen Datentyp gibt, brauchen Sie sich darüber keine Gedanken zu machen.

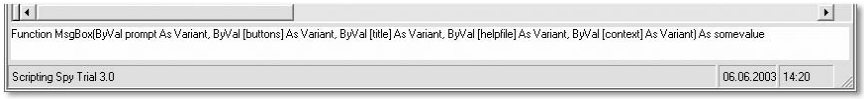

Abbildung 1.9 Die Parameterinformationen des Befehls werden im unteren Teil des Editors angezeigt.

5. Alles, was Sie nun tun müssen, ist den ersten Parameter anzugeben. Damit definieren Sie den auszugebenden Text. Alle weiteren Parameter sind optional, können also zunächst entfallen. Der Cursor sollte hinter dem Wort `MsgBox` stehen. Geben Sie nun ein Leerzeichen und dann den auszugebenden

Text ein, den Sie aber in Anführungszeichen oben einfassen müssen. Das komplette Listing sollte am Ende wie folgt aussehen:

```
' ************************************************
' *
' * VBScript Quelldatei -- erstellt mit Scripting Spy 3.0
' *
' *
' *
' * HINWEISE: Mein erstes Skript - es gibt "Hallo Welt" aus.
' *
' ************************************************
MsgBox "Hallo Welt"
```

1.4.4 Das Skript speichern und ausführen

Sie können das Skript nun speichern und ausführen. Wählen Sie dazu einfach **Datei • Speichern** aus dem Menü aus und geben Sie einen Namen Ihrer Wahl in das Feld **Dateiname** ein.

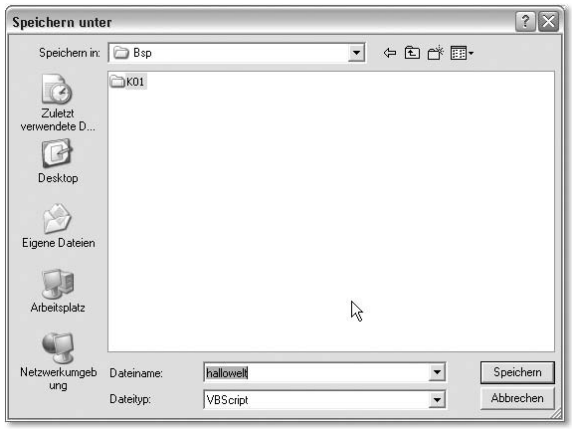

Abbildung 1.10 Speichern der Datei

Hinweis Scripting Spy wählt als Dateityp automatisch VBScript aus und vergibt die notwendige Dateinamenserweiterung **.vbs**. Bei anderen Editoren müssen Sie eventuell selbst auf eine korrekte Namensvergabe achten. Steht VBScript als Dateityp nicht zur Verfügung, wählen Sie Textdateien, Nur-Text oder ASCII-Text aus und geben die Dateinamenserweiterung **.vbs** nach dem Dateinamen selbst an. Verwenden Sie Notepad oder den Windows-Editor, müssen Sie als Dateityp **Alle Dateien** auswählen, damit nicht an die Endung **.vbs** auch noch ein **.txt** angehängt wird.

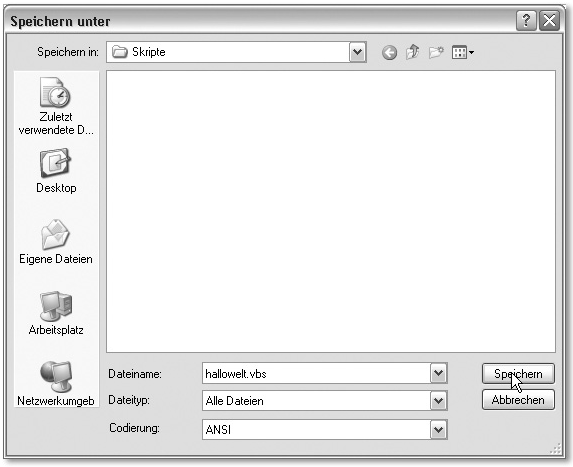

Abbildung 1.11 Notwendige Einstellungen beim Speichern mit dem Windows-Editor

Ist die Datei gespeichert, können Sie sie ausführen. Dazu gibt es mehrere Möglichkeiten. Scripting Spy bietet beispielsweise die Möglichkeit, die Datei direkt aus dem Editor zu starten, indem Sie die Schaltfläche **Skript ausführen** anklicken.

Abbildung 1.12 Das aktuelle Skript ausführen

Nicht jeder Editor bietet jedoch diese Möglichkeit, und vor allem können Sie so keinen Parameter an das Skript übergeben oder es mit einem bestimmten Skripthost ausführen. Wenn Sie nicht auf die Möglichkeiten Ihres Editors beschränkt sein möchten, können Sie wie folgt verfahren.

1.4.5 Ein Skript per Kontextmenü oder Doppelklick ausführen

Am einfachsten lässt sich ein Skript per Doppelklick auf die Skriptdatei starten. Dazu öffnen Sie das Verzeichnis, in dem sich die Datei befindet, einfach im Explorer oder Arbeitsplatz und klicken doppelt mit der Maus darauf. Sie wird dann standardmäßig mit dem Skripthost **WScript.exe** ausgeführt, der Ausgaben beispielsweise in Dialogfeldern ausgibt. Dies lässt sich jedoch ändern. Alternativ können Sie auch **CScript.exe** verwenden. Dabei handelt es sich um die Kommandozeilenversion. Ein- und Ausgaben erfolgen daher in der Kom-

mandozeile des Betriebssystems im DOS-Fenster beziehungsweise in der Eingabeaufforderung. Wenn Sie nicht generell alle Skripte mit **CScript.exe** ausführen möchten, sollten Sie den Standard-Skripthost nicht ändern. Statt dessen können Sie einzelne Skripte wahlweise mit dem einen oder anderen Host ausführen, indem Sie die Kontextmenüeinträge einer Skriptdatei verwenden.

Klicken Sie auf **Öffnen,** wird das Skript mit dem Standard-Skripthost (in der Regel **WScript.exe**) ausgeführt. Es passiert also das Gleiche wie bei einem Doppelklick auf die Datei. Alternativ können Sie über den Eintrag **Mit Konsole öffnen** das Skript zur Ausführung an den Host **CScript.exe** übergeben. Falls **CScript.exe** der Standard-Skripthost ist, können Sie das Skript mit dem Eintrag **Öffnen mit Eingabeaufforderung** in der Kommandozeile ausführen. Der Eintrag ist nun aber Standardeintrag des Kontextmenüs und steht ganz oben. Mit dem Eintrag **Öffnen**, der weiter unten steht, führen Sie das Skript dann mit **Wscript.exe** aus.

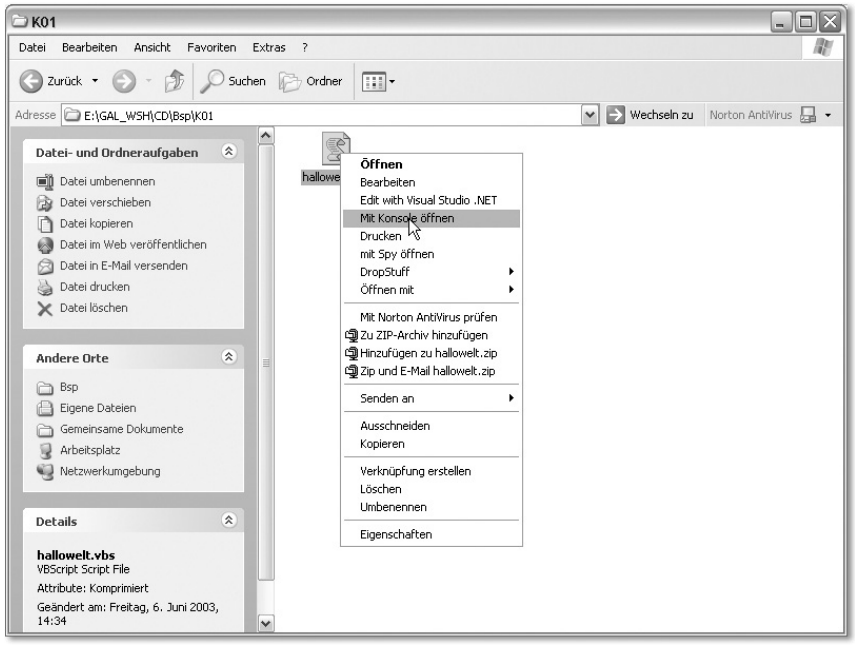

Abbildung 1.13 Ein Skript über das Kontextmenü starten

1.4.6 Skripte auf Kommandozeilenebene starten

Alternativ können Sie Skripte auch per Kommandozeile starten. Entweder wählen Sie dazu **Start • Ausführen** aus dem Startmenü von Windows aus und geben dann den entsprechenden Befehl ein. Das Fenster schließen Sie dann mit **OK**. Oder Sie öffnen die Kommandozeile von Windows und führen dort den

Befehl aus. Geben Sie ihn dazu am Prompt ein und schließen Sie ihn mit ↵ ab. In beiden Fällen ist der Befehl der gleiche. Er hat folgende Syntax:

```
Host Skriptname [Optionen][Skriptparameter]
```

Als Host geben Sie **WScript.exe** oder **CScript.exe** an, je nachdem, mit welchem Skripthost Sie das Skript ausführen möchten. Danach geben Sie den Namen des Skriptes einschließlich Pfad an. Der Pfad muss aber nur angegeben werden, wenn sich das Skript nicht im aktuellen Verzeichnis befindet. Optional können Parameter an das Skript übergeben werden, die dann hinter dem Skriptnamen folgen. Mit Hilfe von Optionen können Sie die Ausführung von Skripten steuern und den WSH einstellen. Wenn Sie also das Skript **HalloWelt.vbs** mit **CScript.exe** ausführen möchten, geben Sie ein:

```
cscript.exe e:\GAL_WSH\Bsp\K01\Hallowelt.vbs
```

Den Pfad zum Skript müssen Sie natürlich an Ihre Verzeichnisstruktur anpassen.

1.5 Optionen zur Konfiguration einsetzen

Wenn Sie Skripte in der Kommandozeile ausführen, haben Sie die Möglichkeit, verschiedene Optionen für die Ausführung zu setzen. Die komplette Liste der Optionen finden Sie in der folgenden Tabelle. Wichtig sind davon vor allem die Optionen `//D` und `//X`, die Ihnen die Fehlersuche erleichtern. Mit der Option `//D` schalten Sie das aktive Debuggen ein. Das heißt, das Skript wird im Skript-Debugger ausgeführt, falls ein Fehler auftritt. Mit der Option `//X` können Sie explizit festlegen, dass das Skript im Debugger ausgeführt werden soll, auch wenn kein Fehler auftritt.

Ein Debugger ist ein Programm, das bei der Fehlersuche im Quellcode hilft. Damit ist es möglich, den Code Schritt für Schritt auszuführen und aktuelle Werte von Variablen zu ermitteln. WSH-Skripte können Sie dem Microsoft-Skript-Debugger debuggen. Er ist auf den meisten Rechnern schon vorhanden. Wenn er nicht verfügbar ist, können Sie ihn von der Skripting-Webseite herunterladen: **www.microsoft.com/germany/scripting**

Option	Beschreibung
`//B`	Batch-Modus: zeigt keine Skriptfehler und Aufforderungen an
`//D`	Aktiviert aktives Debuggen
`//H:CScript`	Ändert den Standard-Skripthost auf **CScript.exe**

Tabelle 1.2 Verfügbare Optionen zur Konfiguration des WSH

Option	Beschreibung
//H:WScript	Ändert den Standard-Skripthost auf **WScript.exe**
//I	Interaktiver Modus (Standard; Gegenteil von //B)
//Job:xxxx	Führt einen WSF-Auftrag aus. Näheres dazu finden Sie in Kapitel 7, Quellcode wieder verwenden.
//Logo	Zeigt das Logo an (Standard) – Gegenteil von //Nologo
//Nologo	Zeigt kein Logo an: Bei Ausführung wird kein Banner angezeigt.
//S	Speichert die aktuellen Befehlszeilenoptionen für den aktuellen Benutzer.
//T:nn	Timeout in Sekunden: maximale Zeit, die ein Skript laufen darf. Die Zeit in Sekunden geben Sie anstelle von nn an.
//X	Führt das Skript im Debugger aus
//U	Verwendet Unicode für umgeleitete Eingabe-/Ausgabe-Vorgänge von der Konsole aus. Gilt nur für Windows 2000 und höher sowie für Windows NT.

Tabelle 1.2 Verfügbare Optionen zur Konfiguration des WSH (Forts.)

Alle Optionen bis auf //H beziehen sich nur auf das auszuführende Skript und werden daher anschließend nicht auch für andere Skripte verwendet. Sie können aber mit Hilfe der Option //S dafür sorgen, dass die gewählten Einstellungen immer verwendet werden, wenn Sie erneut ein Skript ausführen. Die Einstellungen werden dann nämlich für den aktuell angemeldeten Benutzer gespeichert. Möchten Sie beispielsweise alle Skripte mit aktivem Debuggen ausführen, geben Sie einmalig bei einem Skript Ihrer Wahl //D //S an.

```
wscript.exe e:\GAL_WSH\Bsp\K01\hallowelt.vbs //D //S
```

Die Option //D wird dann für den aktuellen Benutzer gespeichert und fortan verwendet. Unter Windows 98/ME muss jedoch Windows so eingerichtet sein, dass eine Benutzeranmeldung beim Start von Windows erforderlich ist.

1.6 Skripte mit WSH-Dateien konfigurieren

Bestimmte Einstellungen können Sie aber auch einem ganz spezifischen Skript zuweisen, indem Sie für das Skript eine WSH-Datei erstellen. Diese wird automatisch im gleichen Verzeichnis wie die Skriptdatei gespeichert und hat auch den gleichen Namen. Nur die Dateinamenserweiterung weicht ab, sie lautet zwingend **.WSH**.

Die WSH-Datei ist ähnlich wie eine INI-Datei für Anwendungen eine einfache Textdatei, die die Einstellungen definiert. Sie brauchen sie allerdings nicht

mühsam mit einem Editor zu erstellen, sondern können dazu ein Dialogfeld nutzen. Rufen Sie einfach im Kontextmenü der Skriptdatei, die Sie konfigurieren möchten, den Eintrag **Eigenschaften** auf und aktivieren Sie dann die Registerkarte **Skript**. Dort können Sie alle Einstellungen vornehmen und diese mit **OK** speichern. Sehr viele Einstellungen sind das allerdings nicht. Mit dem Kontrollkästchen **Skript nach folgender Anzahl von Sekunden anhalten**, können Sie festlegen, dass für die Ausführung des Skriptes nur eine begrenzte Zeit zur Verfügung steht. Diese geben Sie durch Auswahl der Anzahl Sekunden im Feld **Sekunden** an. Nach der definierten Zeitspanne wird das Skript abgebrochen, wenn es dann immer noch nicht beendet ist. Die zweite Einstellung betrifft die Ausführung des Skriptes in der Kommandozeile von Windows. Wird es mit **CScript.exe** ausgeführt, können Sie wahlweise das WSH-Logo anzeigen lassen, das die Versionsnummer des WSH anzeigt, oder es unterdrücken. Um es zu unterdrücken, deaktivieren Sie das Kontrollkästchen **Logo anzeigen, wenn das Skript in der Befehlskonsole ausgeführt wurde**.

Wenn Sie Einstellungen festgelegt haben, die von den Standardeinstellungen abweichen, wird die WSH-Datei automatisch erstellt, wenn Sie **Übernehmen** oder **OK** anklicken. Wenn Sie die Standardeinstellungen wiederherstellen möchten, klicken Sie dazu einfach die Schaltfläche **Auf Standard zurücksetzen** des Dialogfelds an. Alternativ können Sie auch im Arbeitsplatz oder Explorer die WSH-Datei löschen.

Abbildung 1.14 Erstellen einer WSH-Datei über den Eigenschaften-Dialog der Skriptdatei

Damit die Einstellungen in der WSH-Datei berücksichtigt werden, dürfen Sie nun aber nicht mehr die Skriptdatei ausführen, sondern müssen die WSH-Datei starten. Das funktioniert genauso wie der Start der Skriptdatei, entweder per Doppelklick oder über das Kontextmenü der WSH-Datei. Das Kontextmenü der WSH-Datei berücksichtigt aber nicht die Einstellungen bezüglich des Standard-Skripthosts. Mit **Öffnen** wird immer **Wscript.exe** verwendet und mit **Mit Konsole öffnen** der Host **CScript.exe**. Der Eintrag **Öffnen** ist immer der Standardeintrag des Kontextmenüs. Führen Sie wie bisher die Skriptdatei aus, werden die Einstellungen nicht berücksichtigt, das Skript ist aber dennoch funktionsfähig.

```
[ScriptFile]
Path=E:\GAL_WSH\CD\Bsp\K01\hallowelt.vbs
[Options]
Timeout=10
DisplayLogo=1
```

Hinweis Die WSH-Datei ist mit der Skriptdatei verknüpft. Innerhalb der WSH-Datei, die Sie beispielsweise im Windows-Editor wie eine normale Textdatei öffnen können, stehen der Pfad und Name der Skriptdatei. Wenn Sie also die Skriptdatei nachträglich umbenennen oder in ein anderes Verzeichnis verschieben, müssen Sie die WSH-Datei manuell anpassen oder neu erzeugen, sonst kann die Skriptdatei nicht ausgeführt werden oder es wird unter Umständen die falsche Version des Skriptes ausgeführt. Bei manueller Anpassung der Datei geben Sie einfach den korrekten Dateinamen und Pfad der Skriptdatei hinter `Path=` in der Datei an. Den Inhalt einer WSH-Datei zeigt das folgende Listing.

Wenn Sie die WSH-Datei mit einem Editor bearbeiten möchten, sollten Sie deren Dateinamenserweiterung vorübergehend in **.txt** umbenennen. Dann können Sie die Datei einfach per Doppelklick öffnen. Ansonsten können Sie Anpassungen jedoch auch vornehmen, indem Sie erneut den Eigenschaften-Dialog des Skriptes aufrufen.

1.7 Syntaxbeschreibungen verstehen

Die bis hierhin verwendeten Befehle waren noch recht einfach. In den nachfolgenden Kapiteln begegnen Ihnen jedoch verstärkt Befehle, die etwas komplexer aufgebaut sind. Für diese Anweisungen wird dann auch eine Syntaxbeschreibung genannt, die Sie im Anfang finden. Wenn Sie einmal die Bedeutung der einzelnen Schreibweisen kennen, können Sie so durch einen Blick in den Anhang die notwendigen Informationen ermitteln.

Eine Syntaxbeschreibung besteht im Wesentlichen aus

- kursiven Bestandteilen, das sind Elemente, die Sie durch eigene Werte ersetzen. Diese Werte können konstante Zahlen oder Zeichenketten oder Variablen sein.
- Teilen mit gerader Schrift. Dabei handelt es sich um die Befehlsnamen, die unveränderbar sind, oder um vordefinierte Konstanten, die Sie nutzen können.
- Teilen in eckigen Klammern. Hierbei handelt es sich um optionale Angaben, die Sie auch weglassen können, wenn Sie sie nicht benötigen
- Elementen, die durch einen senkrechten Strich getrennt sind. Dies sind alternative Werte, meist Konstanten, die Sie an dieser Stelle verwenden können.

Werden Parameter in runde Klammern eingefasst, sind diese an der angegebenen Stelle einzugeben.

Zur Verdeutlichung soll die Syntaxbeschreibung der `Split`-Funktion dienen. Sie enthält alle folgenden Elemente:

```
Split(Zeichenkette[, Trennzeichen[, Anzahl[, vbBinaryCompare|
    vbTextCompare]]])
```

An die Funktion können Sie bis zu vier Werte übergeben. Der erste, `Zeichenkette`, ist Pflicht. Sie ersetzen ihn (weil kursiv gesetzt) durch einen beliebigen geeigneten Wert. Der zweite, `Trennzeichen`, ist optional, da er in eckigen Klammern angegeben wird. Aber auch diesen ersetzen Sie, wenn Sie ihn angeben möchten, durch einen beliebigen geeigneten Wert. Die Tatsache, dass vor dem Namen des Wertes ein Komma innerhalb der eckigen Klammern steht, hat natürlich auch eine Bedeutung: Geben Sie den optionalen Wert an, müssen Sie vor ihm auch das Komma angeben. Lassen Sie ihn weg, entfällt auch das Komma.

Für den Fall, dass Sie den zweiten Wert (Parameter genannt) angeben, können Sie auch den dritten, `Anzahl` angeben. Lassen Sie den zweiten Parameter weg, können Sie auch den dritten nicht angeben. Das wird dadurch kenntlich gemacht, dass sich die eckige Klammer für den dritten Parameter innerhalb der eckigen Klammer befindet, die den zweiten Parameter umgibt. Könnten Sie den dritten Parameter auch dann angeben, wenn Sie den zweiten weglassen, sähe die Syntax nämlich wie folgt aus:

```
Split(Zeichenkette[, Trennzeichen][, Anzahl][, vbBinaryCompare|
    vbTextCompare])
```

Der letzte Parameter der Funktion erwartet einen vorgegebenen Wert, kann also nicht frei festgelegt werden. Die beiden möglichen Werte werden in eckige Klammern eingefasst, weil der Parameter ebenfalls optional ist. Sie können also entweder den Wert `vbBinaryCompare` oder den Wert `vbTextCompare` verwenden.

9 Sicherheit

9 Sicherheit

Bekannt geworden ist der Windows Script Host im Wesentlichen durch Melissa! Dieser Virus hat fast 80 Prozent aller Unternehmen und über 70 Prozent aller privaten PCs infiziert. Der Grund lag einfach darin, dass er sich die Fähigkeiten des Windows Script Host zunutze gemacht und auch andere Anwendungen wie Outlook für seine Zwecke missbraucht hat. Wie Sie sich vor solchen Attacken schützen und sicherstellen, dass Ihre Skripte auch von vorsichtigen Anwendern ausgeführt werden können, zeigt das vorliegende Kapitel.

Das Thema »Sicherheit« ist aus mehreren Gründen für alle Skriptprogrammierer relevant. Zum Beispiel deshalb, weil nach Melissa sowohl die Software-Hersteller an der Sicherheit gearbeitet haben als auch viele Anwender vorsichtig geworden sind und sich Virenscanner zugelegt haben. Beides kann dazu führen, dass Ihre Skripte nicht mehr ausgeführt werden können oder nicht mehr fehlerfrei funktionieren. Das heißt, als Entwickler sollten Sie sehr genau wissen, unter welchen Voraussetzungen Ihre Skripte überhaupt ausgeführt werden können. Zudem kommt bei vielen Entwicklern über kurz oder lang die Frage auf, wie sie ihren mühsam erstellten Code vor neugierigen Blicken schützen können.

Aber auch als Anwender dürfte es Sie natürlich interessieren, wie Sie Ihren Rechner trotz WSH vor Viren schützen können, ohne dass Sie dazu auf eigene Skripte verzichten müssen. Wenn Sie Administrator eines kleineren oder größeren Netzwerks sind, haben Sie natürlich ein ganz besonderes Interesse daran, dass nur die von Ihnen abgesegneten Skripte ausgeführt werden können, nicht aber Viren und Würmer, die dann Schaden anrichten. Welche Möglichkeiten es dafür gibt, hängt aber entscheidend von der WSH-Version ab. Der WSH 5.6 bietet die Möglichkeit Skripte zu verschlüsseln, zu signieren und die Ausführung von Skripten auf signierte Skripte zu beschränken.

9.1 Allgemeine Sicherheitstipps und -ratschläge

Zunächst einmal muss man zugeben, dass die WSH-Kritiker Recht haben mit dem Vorwurf, dass der WSH eine Sicherheitslücke darstellt. Erst der WSH macht es schließlich möglich, Programme auch unsichtbar zu steuern. Nur so konnte Melissa sich unbemerkt in E-Mails über Outlook verschicken. Dennoch kann ein Rechner auch mit installiertem WSH weitgehend vor Viren und dem damit verbundenen Datenverlust geschützt werden. Der zweite Unsicherheitsfaktor ist nämlich immer noch der Benutzer.

Und der Benutzer hat es in der Hand, sich entsprechend zu verhalten. Generell sollten Sie folgende Regeln beachten, wenn Sie bei installiertem WSH ohne Virenscanner mit Ihrem Rechner arbeiten. Vorausgesetzt natürlich, Ihnen liegt etwas an Ihren Daten.

- Wenn Sie E-Mails mit Outlook 2000 empfangen oder versenden, installieren Sie das Service Pack 1. Es unterbindet Anhänge, die potenziell Viren enthalten könnten. Das sind neben Access-Datenbanken auch VBScript-Dateien, JScript-Dateien und EXE-Dateien. Solche Anhänge in eingehenden E-Mails werden nicht mehr angezeigt und können so auch nicht ausgeführt werden.
- Nutzen Sie Outlook Express für E-Mails und Newsgroups, sollten Sie die Sicherheitseinstellungen für Outlook Express auf das Maximum heraufsetzen. Damit können Sie verhindern, dass bereits in der HTML-Vorschau der Mails Plug-ins und HTML- und Skript-Code ausgeführt wird, der Viren und Würmern Tür und Tor öffnet.
- Führen Sie Anhänge in E-Mails, die von fremden Leuten kommen oder die von Bekannten kommen, von denen Sie die Anhänge aber nicht erwarten, niemals einfach aus. Bedenklich sind nicht nur Skriptdateien, sondern natürlich auch Office-Dokumente, denn auch sie können WSH-Objekte in VBA ansprechen. Denkbar wäre sogar, dass eine »harmlose« Word-Datei ein Skript erstellt, das Ihre Festplatte formatiert und das dann per Autostart-Eintrag beim nächsten Systemstart ausgeführt wird. Sie sollten also bei jeglichen Anhängen, vor allem natürlich bei EXE-Dateien, ein gesundes Misstrauen an den Tag legen.
- Blenden Sie die Dateinamenserweiterungen für bekannte Dateitypen ein. Nur dann können Sie nämlich eine JPEG-Datei auch sicher von einer Skriptdatei unterscheiden, die beispielsweise **meinscript.jpg.vbs** heißt. Mit ausgeblendeten Dateitypen würden Sie nämlich **meinscript.jpg** angezeigt bekommen. Ein Doppelklick auf die Datei öffnet dann nicht eine Grafikdatei, sondern startet das Skript, das großen Schaden anrichten könnte.
- Stellen Sie für den Internet Explorer die Sicherheit auch für die lokale und Intranetzone niemals auf ganz niedrig. Viren und Würmer könnten sich auch als HTML-Datei einschleichen, die Sie dann vielleicht von der lokalen Festplatte öffnen und damit die Sicherheitseinstellungen für das Internet umgehen.
- Wenn Sie Skriptdateien von anderen bekommen oder sich als Beispiele aus dem Internet herunterladen, sehen Sie sich die Skripte erst an, bevor Sie sie ausführen. Nicht immer verbirgt sich hinter der Datei auch der angekündigte Inhalt.

- Falls Sie Windows NT, Windows 2000 oder Windows XP verwenden, nutzen Sie deren Möglichkeit Benutzer und Benutzerrechte zu verwalten. Skripte werden immer mit den Rechten des aktuell angemeldeten Benutzers ausgeführt. Wenn der keine Rechte zur Manipulation der Registry hat, kann ein bösartiges Skript hier keinen Schaden anrichten. Gleiches gilt natürlich für Dateien, auf die ein Benutzer oder eine Benutzergruppe keinen Zugriff hat. Auch die sind vor Zugriffen per Skript geschützt.

Wenn Sie diese Ratschläge beherzigen, kann Ihnen kaum etwas passieren. Wichtig ist immer: Erst denken, dann klicken. Allerdings sind für einige der Hinweise Änderungen an der Standardkonfiguration von Windows erforderlich. Wie Sie diese Änderungen durchführen, wird daher nachfolgend beschrieben.

9.1.1 Outlook- und Outlook-Express-Sicherheitseinstellungen heraufsetzen

Die meisten Viren und Würmer, die sich des WSH bedienen, schleichen sich über Anhänge von E-Mails oder in HTML-E-Mails ein und nutzen dazu die laschen Sicherheitseinstellungen von Outlook und Outlook Express. Gerade das lässt sich jedoch problemlos ändern, nur wissen viele Anwender leider nicht, dass es geht und dass es überhaupt ein Sicherheitsrisiko gibt.

Wenn Sie Outlook 2000 oder höher verwenden, gehen Sie wie folgt vor, um die Sicherheitseinstellungen zu kontrollieren und bei Bedarf zu korrigieren:

- Starten Sie Outlook und wählen Sie dann **Extras • Optionen** aus dem Menü aus.
- Aktivieren Sie die Registerkarte **Sicherheit**.
- In der Mitte des Dialogs finden Sie eine Gruppe **Inhalt sichern**. Hier sollten Sie sicherstellen, dass im Listenfeld Zone der Eintrag **Eingeschränkte Sites** ausgewählt ist. Dies ist die Zone mit der höchsten Sicherheitsstufe.

  Hinweis Wenn Sie möchten, können Sie die Zone nun noch anpassen beziehungsweise die Einstellungen kontrollieren. Klicken Sie dazu auf **Zoneneinstellungen** und bestätigen Sie die Warnung mit **OK**. In dem eingeblendeten Dialog klicken Sie dann auf **Zone anpassen**. Im Normalfall sind hier aber keine Änderungen mehr notwendig. Diese Zone ist wirklich sicher.

- Schließen Sie den Dialog mit **OK** oder klicken Sie auf **Übernehmen** um geänderte Einstellungen zu speichern.

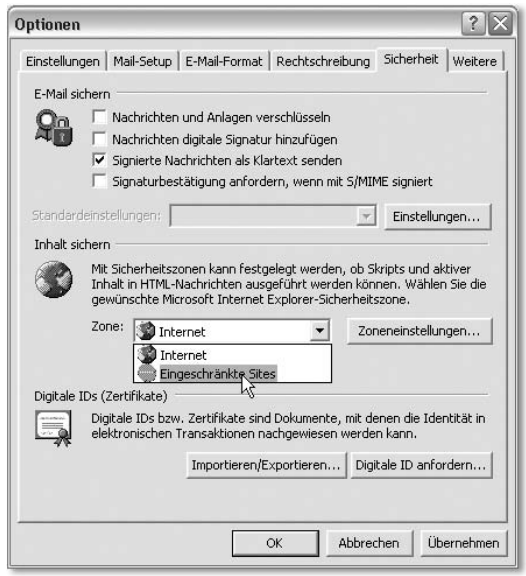

Abbildung 9.1 Sicherheitszone einstellen

Falls Sie Outlook Express nutzen, funktioniert die Anpassung ganz ähnlich.

- Starten Sie Outlook Express und wählen Sie **Extras • Optionen** aus.
- Aktivieren Sie auch hier die Registerkarte **Sicherheit**.
- Ganz oben im Dialogfeld unter **Virenschutz** finden Sie die Option **Zone für eingeschränkte Sites (sicherer)**. Aktivieren Sie diese, wenn sie nicht aktiviert ist.

> **Hinweis** Abhängig von Ihrer Outlook-Express-Version könnte der Dialog auch etwas anders aussehen. Unter Umständen müssen Sie die Zone wie in Outlook aus einem Kombinationslistenfeld auswählen. Wichtig ist nur, dass Sie die Zone für eingeschränkte Sites auswählen.

- Stellen Sie sicher, dass das Kontrollkästchen **Warnung anzeigen, wenn andere Anwendungen versuchen E-Mail unter meinem Namen zu versenden** aktiviert ist.
- Wenn Sie noch mehr Sicherheit möchten, können Sie auch das zweite Kontrollkästchen **Speichern oder öffnen von Anlagen, die möglicherweise Viren enthalten, nicht zulassen** aktiveren. Damit verringern Sie das Risiko, versehentlich gefährliche Anhänge zu öffnen.

> **Hinweis** Ob es diese beiden Kontrollkästchen gibt, hängt von der Outlook-Express-Version ab.

Abbildung 9.2 Die Sicherheitseinstellungen für Outlook Express 6

- Klicken Sie auf **OK** oder **Übernehmen**, um die Einstellungen zu speichern.

> **Tipp** In Outlook Express können Sie die Einstellungen für die Zone nicht einsehen und ändern. Möchten Sie die Einstellungen kontrollieren, geht das nur über den Internet Explorer. Outlook und Outlook Express verwenden nämlich die Sicherheitszonen des Internet Explorers.

9.1.2 Sicherheitseinstellungen des Internet Explorers einstellen

Die Sicherheitseinstellungen des Internet Explorers sind relevant, wenn es darum geht, Ihren Rechner vor Viren und Würmern zu schützen, die sich als Skript oder Link auf einer Website verbergen. Außerdem sind die Einstellungen für die Sicherheitszonen auch für Outlook und Outlook Express von Bedeutung. Um den Internet Explorer einzustellen, gehen Sie wie folgt vor:

- Starten Sie den Internet Explorer.
- Wählen Sie für den Internet Explorer 5 und höher den Menüeintrag **Extras • Internetoptionen** aus. In früheren Internet-Explorer-Versionen finden Sie die Einstellungen im Menü **Bearbeiten**.
- Aktivieren Sie die Registerkarte **Sicherheit**. Ganz oben im Dialogfeld können Sie nun die Zonen markieren, deren Einstellungen Sie festlegen möchten. Wichtig sind vor allem die Zonen **Internet** und **Lokales Intranet**. Letztere wird verwendet, wenn Sie Dateien von Netzwerklaufwerken und Freigaben ausführen.

Tipp Aus Sicherheitsgründen sollten Sie jedoch bei Verwendung des Internet Explorers 4 auf die Version 5.5 oder höher updaten, da in der Version 4.0 noch kein Skripting deaktiviert werden kann und damit ein Schutz vor bösartigem Code, der die WSH-Objekte verwendet, nicht möglich ist.

Tipp Die Zone **Lokales Intranet** wird nicht verwendet, wenn Sie Dateien von der lokalen Festplatte ausführen. Leider können Sie für die Zone Arbeitsplatz, die verwendet wird, wenn Sie Dateien vom lokalen Laufwerk ausführen, keine Einstellungen festlegen. Sie können den Internet Explorer aber austricksen. Geben Sie ein lokales Laufwerk frei, verbinden Sie es mit einem Laufwerksbuchstaben, beispielsweise X:. Sie können dann die Dateien von Laufwerk X aus ausführen oder den Freigabenamen angeben, wenn Sie eine Datei ausführen.

Abbildung 9.3 Wenn Sie eine Datei von einer Freigabe aufrufen, wird die Zone Lokales Intranet verwendet, auch wenn die Freigabe sich auf einem lokalen Laufwerk befindet

- Markieren Sie zunächst das Symbol **Internet** und klicken Sie dann auf **Stufe anpassen**.

Abbildung 9.4 Sicherheitszonen für Internet und Lokales Intranet anpassen

- Wählen Sie nun aus der Pulldown-Liste **zurücksetzen zu** den Eintrag **Mittel** aus.

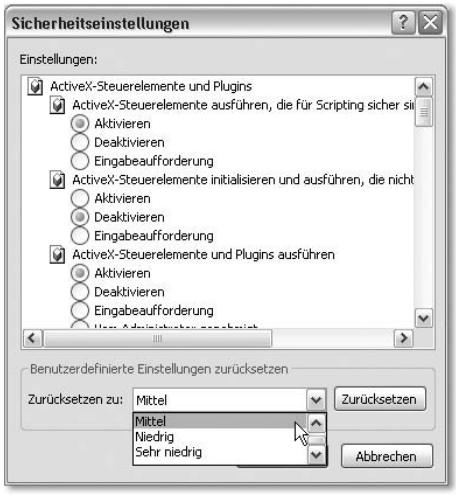

Abbildung 9.5 Auswählen der Sicherheitseinstellungen

> **Hinweis** Falls Sie es noch sicherer haben möchten, können Sie auch **Hoch** auswählen. Dann werden jedoch auch keine Plug-ins wie Flash- und SVG-Viewer mehr ausgeführt.

- Klicken Sie auf **OK**, um die Einstellungen zu übernehmen und zum vorherigen Dialog zurückzukehren.
- Markieren Sie nun das Symbol **Lokales Intranet** und verfahren Sie damit genauso. Auch dafür sollten Sie mindestens die Stufe **Mittel** auswählen.
- Schließen Sie dann den Dialog **Internetoptionen** mit **OK**.

9.1.3 Dateinamenserweiterungen einblenden

Wichtig ist vor allem, dass Sie eine Datei korrekt erkennen, bevor Sie sie ausführen. Standardmäßig zeigt Windows nämlich keine Dateinamenserweiterungen für registrierte Dateitypen an. Als Dateinamenserweiterung wird immer der letzte Teil des Dateinamens nach dem Punkt verwendet. Allerdings hindert Windows Sie nicht daran zwei *Erweiterungen* zu vergeben. Sie können eine JPG-Datei also durchaus **Bild.txt.jpg** nennen. Windows wird dann bei ausgeblendeten Dateinamenserweiterungen **Bild.txt** im Arbeitsplatz anzeigen. Klicken Sie aber doppelt darauf, öffnet sich ein Grafikprogramm und zeigt die Grafik an, vorausgesetzt, die Datei enthält Grafikdaten.

Dieses Verhalten von Windows haben sich Virenprogrammierer zunutze gemacht und versenden ihre Viren und Würmer (oder 0190-Dialer) getarnt als JPG-Dateien, HTML-Dateien oder AVI-Dateien. Dahinter verbergen sich jedoch in Wirklichkeit Skriptdateien oder EXE-Dateien. Outlook Express und Outlook 2003 zeigen zwar automatisch die Dateinamenserweiterungen in den Anlagen an, unabhängig von den Windowseinstellungen, frühere Versionen machen das aber nicht. Das Problem wird noch größer, wenn sich die Dateien erst einmal auf der Festplatte befinden, beispielsweise nach einem Download oder nachdem Sie sie aus der E-Mail gespeichert haben.

Wenn Sie beispielsweise eine Grafikdatei **testbild.bmp** und eine Skriptdatei **testbild.bmp.vbs** erstellen und diese Dateien mit den ausgeblendeten Dateinamenserweiterungen im Arbeitsplatz ansehen, sieht das wie in Abbildung 9.6 aus. Der gut gewählte Name mit dem Text »bild« lässt den Schluss zu, beides seien Bilder. Das untere ist jedoch die VBScript-Datei. Erkennbar ist das für den unbedarften Anwender nur daran, dass sich die Symbole unterscheiden. Viele Benutzer stutzen dann aber leider nicht. Sie gehen davon aus, dass Windows so viele Symbole hat, das wird schon richtig sein. Ein Doppelklick auf die untere Datei würde das Skript ausführen und nicht die Grafik öffnen. Ein bösartiges Skript könnte damit schon die Festplatte formatieren.

Abbildung 9.6 Eine Grafikdatei und eine getarnte Skriptdatei im Arbeitsplatz

Eine bessere Chance, solche Tretminen zu enttarnen, haben Sie auf jeden Fall, wenn Sie sich alle Dateinamenserweiterungen anzeigen lassen. Dazu gehen Sie wie folgt vor:

- Öffnen Sie ein Verzeichnis Ihrer Wahl im Arbeitsplatz.
- Wählen Sie **Extras • Ordneroptionen** aus.
- Klicken Sie auf die Registerkarte **Ansicht**.
- Deaktivieren Sie das Kontrollkästchen **Erweiterungen bei bekannten Dateitypen** ausblenden.

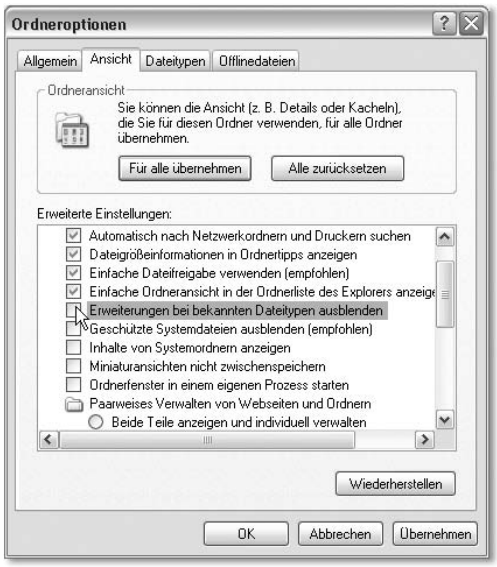

Abbildung 9.7 Dateinamenserweiterungen einblenden

- Klicken Sie auf die Schaltfläche **Für alle übernehmen**. Sie weisen damit die Einstellungen nicht nur dem gerade geöffneten Ordner, sondern allen Ordnern zu.
- Schließen Sie den Dialog mit **OK**.

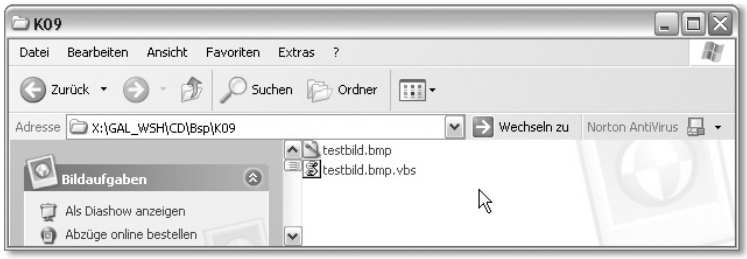

Abbildung 9.8 So lässt sich sehr viel einfacher feststellen, was wahre Grafiken und was getarnte Skripte sind.

9.1.4 Benutzer einrichten und verwalten

Benutzer einzurichten und deren Rechte zu bestimmen ist eine sehr einfache Methode, den Rechner sicherer zu machen. Über die Benutzerrechte legen Sie genau fest, was ein Benutzer darf und was nicht, und dadurch auch die Möglichkeiten eines Skriptes. Das nützt aber nur etwas, wenn Sie sich hinterher auch als Benutzer mit eingeschränkten Rechten und nicht als Administrator am

System anmelden. Zudem setzt dies Windows NT, Windows 2000 oder Windows XP als Betriebssystem voraus.

Tipp Sinnvoll ist die Einrichtung eines Benutzers mit eingeschränkten Rechten vor allem dann, wenn sich wichtige Daten auf Ihrem Rechner befinden und Sie den Rechner mit anderen Benutzern teilen müssen, denen Sie obige Verhaltensregeln nicht zumuten können oder die sich daran nicht halten. Sie stellen mit einem eingeschränkten Benutzer dann außerdem sicher, dass das System nicht durch fehlerhafte Einstellungen ruiniert wird.

Die Vorgehensweise zur Einrichtung von Benutzern hängt davon ab, welche Windows-Version Sie verwenden und welche Netzwerkeinstellungen vorliegen. Nutzen Sie einen Domänenserver und melden sich an die Windows-2000- oder NT-Domäne an, werden die Benutzer auf dem Server verwaltet. In diesem Fall müssen Sie auch dort die Benutzerrechte festlegen beziehungsweise vom Server-Administrator festlegen lassen. Verfügen Sie nicht über ein Netzwerk oder melden sich zumindest nicht an eine Domäne an, verwalten Sie die Benutzer auf Ihrem lokalen Rechner. Für einen solchen Fall gilt die folgende Beschreibung. Hier wird Windows XP Professional eingesetzt, die *sichere Anmeldung* ist aktiviert. Für Windows NT und Windows 2000 können sich daher kleinere Abweichungen ergeben. Bei aktivierter schneller Anmeldung in Windows XP

- starten Sie die Benutzerverwaltung von Windows. Bei Windows XP wählen Sie dazu **Start • Alle Programme • Systemsteuerung** aus und klicken doppelt auf **Benutzerkonten**.
- Aktivieren Sie die Registerkarte **Erweitert** und klicken dort die Schaltfläche **Erweitert an**.
- Es erscheint nun ein Dialogfeld **Lokale Benutzer und Gruppen**, in dem alle aktuell eingerichteten Benutzer aufgeführt werden. Hier können Sie einen neuen Benutzer hinzufügen, indem Sie zunächst den Ordner **Benutzer** anklicken. In der rechten Dialoghälfte werden dann alle Benutzer aufgeführt, die bereits eingerichtet sind.
- Wählen Sie nun **Aktion • Neuer Benutzer** aus. Es erscheint ein Dialogfeld, in das Sie die Benutzerdaten eingeben können. Der **Benutzername** ist dabei der Name, den Sie für die Anmeldung eingeben. Im Feld **Vollständiger Name** können Sie auch Titel, Vorname etc. angeben. Standardmäßig ist das Kontrollkästchen Benutzer muss Kennwort bei der nächsten Anmeldung ändern aktiviert. Das heißt, dass Sie (oder der Benutzer des Kontos) bei der ersten Anmeldung aufgefordert werden, das Kennwort festzulegen. Wenn Sie die Option aktiviert lassen, gilt das festgelegte Kennwort nur für die erste Anmeldung. Deaktivieren Sie diese Option, haben Sie jedoch die Möglich-

keit, das Kennwort für unbegrenzte Zeit festzulegen, indem Sie das Kontrollkästchen **Kennwort läuft nie ab** aktivieren. Klicken Sie anschließend auf **Erstellen** (siehe Abbildung 9.9).

Hinweis Um Benutzer einzurichten oder Benutzerrechte zu verändern, benötigen Sie Administratorrechte, müssen sich also als Administrator anmelden.

Der Benutzer wird nun erstellt und automatisch der Benutzergruppe **Benutzer** zugeordnet. Das ist zwar eine gute Einstellung, noch sicherer können Sie es aber machen, indem Sie eine andere Gruppenzuordnung verwenden. Sie können den Dialog nun mit **Datei • Beenden** schließen.

Abbildung 9.9 Festlegen der Benutzereigenschaften und Erstellen des Kontos

Hinweis Der neue Benutzer bringt natürlich nur etwas, wenn Sie (oder der spätere Benutzer des Rechners), sich auch damit anmelden. Nach einem Neustart des Rechners oder nachdem Sie **Start • Abmelden** ausgewählt haben, müssen Sie dazu den derzeit verwendeten Benutzernamen durch den Namen des eben eingerichteten Benutzers ersetzen und im Passwort-Feld das Kennwort für den Benutzer angeben.

9.2 Die Sicherheit mit Virenscannern erhöhen

Zusätzlich oder alternativ zu den oben beschriebenen Maßnahmen erhöhen natürlich auch gute Virenscanner die Sicherheit. Gute, professionelle Tools prüfen nicht nur auf Viren, sondern kontrollieren auch Skriptaktionen, indem sie immer dann Alarm schlagen, wenn gefährliche Objekte oder Methoden wie beispielsweise das `WSHShell`-Objekt verwendet werden.

Egal, welchen Virenscanner Sie verwenden möchten, Sie sollten unbedingt auf folgende Punkte achten:

- Der Virenscanner darf auf keinen Fall jegliche Skriptausführung unterbinden, denn dann können Sie Ihre WSH-Skripte nicht mehr ausführen.
- Er sollte Skriptdateien aber dennoch prüfen, Ihnen jedoch die Wahl überlassen, wie damit zu verfahren ist.
- Der Virenscanner beziehungsweise die Virendefinitionsdateien des Scanners sollten immer auf dem neuesten Stand sein, beispielsweise durch eine regelmäßige Online-Aktualisierung.
- Er sollte auch Zugriffe auf Dateien überwachen, die über einen Netzwerkrechner erfolgen.

Sehr gut ist Norton Antivirus bei der Überwachung von Skripten. Hier haben Sie bei der Ausführung von Skripten, die gefährliche Anweisungen enthalten, die Möglichkeit die Vorgehensweise zu bestimmen.

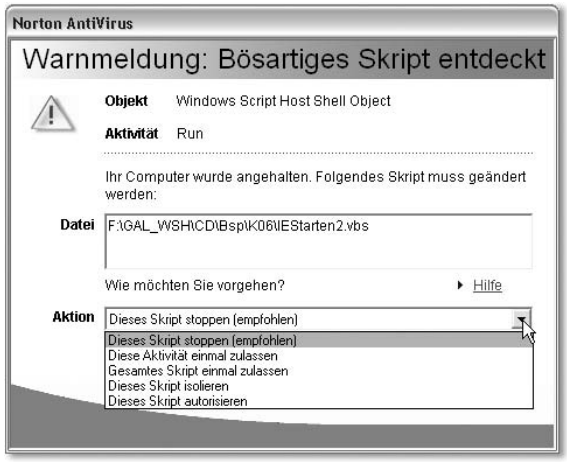

Abbildung 9.10 Möglichkeiten von Norton Antivirus

Wenn Sie nicht möchten, dass das Skript ausgeführt wird, wählen Sie die empfohlene Option **Dieses Skript stoppen (empfohlen)** aus. Dann wird das Skript gestoppt. Falls Sie allerdings den Debugger aktiviert haben, wird er im Anschluss gestartet.

Die Auswahl **Diese Aktivität einmal zulassen** ermöglicht, dass das Objekt erstellt wird, das für den Aufruf des Virenscanners gesorgt hat. Wird es allerdings mehrfach im Skript neu erzeugt, erscheint jedes Mal wieder die Meldung des Virenscanners.

Bei Auswahl von **Gesamtes Skript einmal zulassen** wird das Skript einmal ausgeführt. Alle Objekte werden ohne weitere Nachfrage erzeugt. Beim nächsten Mal meldet sich der Virenscanner aber wieder.

Wenn Sie sicherstellen möchten, dass das Skript nie wieder ausgeführt wird, wählen Sie dazu **Dieses Skript isolieren** aus. Sollte es Ihnen allerdings versehentlich passieren, dass Sie diese Option auswählen, lässt sich das nicht wieder rückgängig machen. Um das Skript erneut starten zu können, erzeugen Sie eine neue, leere Skriptdatei, kopieren dann den Code der isolierten Datei in die neue Skriptdatei und löschen anschließend die alte Datei. Die neue Datei lässt sich dann wieder ausführen.

Falls die Meldungen des Virenscanners Sie stören, wenn Sie Ihre eigenen Skripte ausführen, können Sie das Skript autorisieren, indem Sie **Dieses Skript autorisieren** auswählen. Das Skript wird nachfolgend ohne weitere Meldung des Virenscanners ausgeführt, solange Sie den Quellcode nicht verändern. Nach einer Änderung müssen Sie das Skript erneut autorisieren.

9.3 Das Sicherheitskonzept des WSH 5.6

Das Sicherheitskonzept des WSH soll dazu dienen, Ihren Rechner vor Viren und Würmern zu schützen, ohne dabei die Funktionalität des WSH einzuschränken. Damit das Sicherheitskonzept in Kraft tritt, müssen Sie es aber explizit aktivieren und Ihre Skripte entsprechend bearbeiten. Die Hauptbestandteile des Sicherheitskonzepts sind:

- Signierung von Skripten
- Verifizierung von Skripten
- Sperren der Ausführung unsignierter Skripte

9.3.1 Skripte signieren

Die Signierung von Skripten dient dazu, die Herkunft eines Skriptes auszuweisen. Der Benutzer kann an der Signatur erkennen, wer es erstellt und veröffentlicht hat, und so entscheiden, ob er der Quelle vertraut. Die Signatur wird am Ende des Skriptes verschlüsselt eingefügt. Um sicherzustellen, dass ein ursprünglich aus einer vertrauenswürdigen Quelle stammendes Skript nicht nachträglich manipuliert wird, wird die Signatur ungültig, sobald das Skript nachträglich verändert wird.

Tipp Eine Signatur sollten Sie immer erst dann hinzufügen, wenn das Skript komplett fertig ist und nicht mehr verändert wird.

Damit Sie eine Skriptdatei signieren können, benötigen Sie zwingend ein Zertifikat. Das können Sie entweder bei Ihrem Administrator erhalten, wenn Ihre Firma eigene Zertifikate besitzt, oder Sie müssen sich dazu an eine öffentliche Zertifizierungsstelle wenden.

Abbildung 9.11 So sieht ein signiertes Skript aus.

Tipp Wenn Sie nur Ihren eigenen Rechner schützen möchten und Ihre signierten Skripte nicht weitergeben möchten, können Sie auch ein Zertifikat erstellen, das nicht öffentlich (das heißt von einer Zertifizierungsstelle herausgegeben) ist und somit für andere Anwender nicht überprüfbar ist. Sie können damit jedoch Ihre eigenen Skripte signieren und dann den WSH so einstellen, dass nur signierte Skripte ausgeführt werden können. Voraussetzung ist dafür aber, dass Sie Office 2000 oder Office XP installiert haben. Damit wird nämlich ein Tool **SELFCERT.EXE** installiert, mit dem Sie digitale Zertifikate erstellen können. Suchen Sie einfach nach einem Programm dieses Namens auf Ihrem Rechner und führen Sie es per Doppelklick aus, um ein Zertifikat zu erstellen.

Haben Sie ein Zertifikat erstellt oder von Ihrem Administrator zur Verfügung gestellt bekommen, können Sie Ihre Skripte damit signieren. Sie müssen dazu ein Skript erstellen, das mit Hilfe der `SignFile`-Methode des `Signer`-Objekts ein Skript Ihrer Wahl signiert. Das `Signer`-Objekt erzeugen Sie ganz einfach mit der `CreateObject`-Methode des `WScript`-Objekts aus der Klasse `Scripting.Signer`. Das Skript ermittelt zunächst, ob die benannten Parameter Zerti-

fikat und Dateiname übergeben wurden. Nur wenn beide vorhanden sind, werden deren Werte in Variablen gespeichert und das `Signer`-Objekt erzeugt. Probleme können allerdings auftreten, wenn die angegebene Datei nicht vorhanden ist. Sie sollten daher unbedingt prüfen, ob es die Datei gibt.

Die `SignFile`-Methode benötigt drei Parameter:

objSigner.SignFile (*Dateiname*, *Zertifikat*, *Zertifikspeicher*)

Als ersten Parameter geben Sie den vollständigen Dateinamen einschließlich Pfad an, als zweiten den Namen des zu verwendenden Zertifikats und als letzten den Zertifikatspeicher. Darunter wird der Ordnername im Verzeichnis **Anwendungsdaten\Microsoft\SystemCertificates** des Benutzers verstanden, in dem die Zertifikate gespeichert sind. Der Standardordnername lautet ***My***, auch für deutsche Windows-Versionen. In diesem Ordner werden alle Zertifikate gespeichert, die mit persönlichen Schlüsseln erstellt wurden. Darunter fallen auch die Zertifikate, die **SELFCERT.EXE** erstellt.


```
'Skriptname: signieren.vbs
'Autor: Helma Spona
'Datum: 11.August 2003
'Beschreibung: signiert ein Skript
'--------- Variablen ------------
Dim strZertifikat
Dim strSkriptname
Dim strZertSpeicher
Dim objSigner
Dim objFSO
'-------- Anweisungen -----------
   If (WScript.Arguments.Named.Exists( _
      "Zertifikat")) and ( _
      WScript.Arguments.Named.Exists( _
      "Dateiname"))  Then
      Set objSigner = _
         WScript.CreateObject( _
         "Scripting.Signer")
      Set objFSO=WScript.CreateObject(_
         "Scripting.FileSystemObject")
      strSkriptname  = _
         WScript.Arguments.Named("Dateiname")
      If objFSO.FileExists(strSkriptname)= _
         false then
         MsgBox "Das angegebene Skript " & _
```

```
            "existiert nicht",vbError
         WScript.Quit
      End If
      strZertifikat  = _
         WScript.Arguments.Named("Zertifikat")
      strZertSpeicher="My"
      objSigner.SignFile strSkriptname, _
         strZertifikat, strZertSpeicher
      MsgBox "Skript ist signiert!"
   Else
      Wscript.Quit
   End If
```

Wenn Sie mit Hilfe dieses Skriptes ein anderes Skript signieren möchten, müssen Sie das Skript wie folgt aufrufen:

```
F:\GAL_WSH\CD\Bsp\K09\signieren.vbs /Zertifikat:"Helma Spona" /
Dateiname:F:\GAL_WSH\CD\Bsp\K08\HalloWelt.vbs
```

Sowohl die Verzeichnisangaben für das Skript **signieren.vbs** als auch den Namen und den Pfad des zu signierenden Skriptes müssen Sie natürlich anpassen. Das gilt auch für den Namen des Zertifikats; hier heißt es »Helma Spona« und wurde mit **SELFCERT.EXE** erstellt.

Tipp Achten Sie unbedingt darauf, Parameterwerte, die Leerzeichen enthalten, wie hier der Zertifikatname, in Anführungszeichen einzufassen.

9.3.2 Skripte verifizieren

Wenn Sie den WSH entsprechend eingerichtet haben, wie dies im Abschnitt 9.3.4, Die Sicherheit des Windows Script Host aktivieren, noch näher erläutert wird, werden nur Skripte mit gültigen Signaturen ausgeführt. Sie können jedoch auch manuell prüfen, ob die Signatur gültig ist. Wie das geht, zeigt das folgende Skript. Es erwartet als Parameter das auszuführende Skript und führt dieses mit der `Run`-Methode des `WSHShell`-Objekts aus, sofern die Signatur gültig ist. Eine Signatur überprüfen Sie mit der `VerifyFile`-Methode des `Signer`-Objekts. Ihr übergeben Sie einfach nur den Dateinamen und einen booleschen Wert, der bestimmt, ob der WSH ein Dialogfeld mit den Signaturinformationen anzeigen soll oder nicht. Geben Sie `False` an, gibt die Methode nur einen booleschen Wert mit dem Ergebnis der Prüfung zurück, zeigt aber keine Signaturinfos an. Die Syntax der `VerifyFile`-Methode lautet:

```
objSigner.VerifyFile (Dateiname, Dialoganzeigen)
```

In dieser ersten Variante des Skriptes wird der Rückgabewert der Methode in einer Variablen `blnErg` gespeichert und abhängig von deren Wert dann das Skript ausgeführt oder eine Fehlermeldung angezeigt.

Abbildung 9.12 Diese Meldung zeigt das Skript an, wenn ein signiertes Skript nachträglich geändert wurde

```
'Skriptname: verifizieren.vbs
'Autor: Helma Spona
'Datum: 11. August 2003
'Beschreibung: führt nach Prüfung
'              der Signatur ein
'              Skript aus
'--------- Variablen ------------
Dim strSkriptname
Dim blnErg
Dim objSigner
Dim objFSO
Dim objWSHShell
'-------- Anweisungen -----------
   If WScript.Arguments.Named.Exists( _
      "Dateiname")  Then
      Set objSigner = _
         WScript.CreateObject( _
         "Scripting.Signer")
      Set objFSO=WScript.CreateObject(_
         "Scripting.FileSystemObject")
      strSkriptname  = _
         WScript.Arguments.Named("Dateiname")
      If objFSO.FileExists(strSkriptname)= _
         false then
         MsgBox "Das angegebene Skript " & _
            "existiert nicht",vbError
         WScript.Quit
      End If
      blnErg=objSigner.VerifyFile(strSkriptname,False)
```

```
      If blnErg=True Then
         Set objWSHShell= _
            WScript.CreateObject("WScript.Shell")
         objWSHShell.Run strSkriptname
      Else
         MsgBox "Die Signatur ist nicht mehr " & _
            "gültig!",vbInformation
      End If
   Else
      Wscript.Quit
   End If
```

Alternativ können Sie aber auch dem Benutzer die Wahl lassen, ob er das Skript trotz ungültiger Signatur ausführen möchte. Dazu übergeben Sie als zweiten Parameter `True` an die `VerifyFile`-Methode (siehe Abbildung 9.13).

```
'Skriptname: verifizieren2.vbs
'Autor: Helma Spona
'Datum: 11. August 2003
'Beschreibung: führt nach Prüfung
'              der Signatur ein
'              Skript aus
...
'-------- Anweisungen -----------
...
      If objFSO.FileExists(strSkriptname)= _
         false then
         MsgBox "Das angegebene Skript " & _
            "existiert nicht",vbError
         WScript.Quit
      End If
      objSigner.VerifyFile strSkriptname,True
   Else
      Wscript.Quit
   End If
```

Wenn Sie ein Zertifikat mit **SELFCERT.EXE** erstellt haben, erscheint auch bei gültigen Signaturen eine Warnung (siehe Abbildung 9.14).

Abbildung 9.13 Diese Warnung erscheint, wenn Sie als Parameter True angeben und die Signatur nicht mehr gültig ist.

Abbildung 9.14 Diese Meldung erscheint bei korrekten Signaturen mit Zertifikaten, die nicht als vertrauenswürdige Stammzertifikate erstellt wurden.

9.3.3 Zertifikate zur den vertrauenswürdigen Stammzertifikaten hinzufügen

Wenn Sie Ihr mit **SELFCERT.EXE** erstelltes Zertifikat zu den vertrauenswürdigen Stammzertifikaten hinzufügen möchten, geht dies mit Hilfe des Internet Explorers. Sie exportieren dazu das Zertifikat in eine Zertifikatdatei und importieren es dann in die Liste der vertrauenswürdigen Stammzertifikatsstellen. Gehen Sie dazu wie folgt vor:

1. Starten Sie den Internet Explorer und wählen Sie **Extras • Internetoptionen** aus.
2. Aktivieren Sie die Registerkarte **Inhalte** und klicken Sie dort auf **Zertifikate**.

3. Auf der Registerkarte **Eigene Zertfikate** werden alle mit **SELFCERT.EXE** erstellten Zertifikate angezeigt. Markieren Sie dasjenige, das Sie als vertrauenswürdiges Stammzertifikat ausweisen möchten.
4. Klicken Sie auf **Exportieren** und in den ersten beiden Dialogen des gestarteten Assistenten auf **Weiter**.
5. Aktivieren Sie nun die Option **Syntaxstandard kryptografischer Meldungen – »PKCS #7«-Zertfikate (.P7B)** und aktivieren Sie dann das Kontrollkästchen **Wenn möglich, alle Zertfikate im Zertifizierungspfad einbeziehen**. Danach wechseln Sie mit **Weiter** zum nächsten Schritt des Assistenten.

Abbildung 9.15 Das Exportformat für das Zertifikat auswählen

6. Im nächsten Schritt bestimmen Sie die Datei, in die das Zertfikat exportiert werden soll. Über den **Durchsuchen**-Button können Sie ein Verzeichnis auswählen beziehungsweise ein neues Verzeichnis dafür anlegen und den Dateinamen angeben. Als Dateinamenserweiterung geben Sie **.P7B** an. Klicken Sie nach Auswahl beziehungsweise Eingabe des Dateinamens auf **Weiter** und im nächsten Dialog auf **Fertig stellen** (siehe Abbildung 9.16).
7. Wenn Sie auf diese Weise das Zertifikat exportiert haben, können Sie es nun importieren. Aktivieren Sie dazu die Registerkarte **Vertrauenswürdige Stammzertifizierungsstellen** und klicken Sie dort auf **Importieren**.
8. Im ersten Schritt des Assistenten klicken Sie auf **Weiter**, im zweiten können Sie über den **Durchsuchen**-Button die Datei auswählen, die Sie soeben exportiert haben. Als Dateityp müssen Sie dazu im Öffnen-Dialog **Alle Dateien (*.*)** oder **»PKCS #7«-Zertifikate (*.spc; *.p7b)** auswählen, damit die Datei angezeigt wird (siehe Abbildung 9.17).

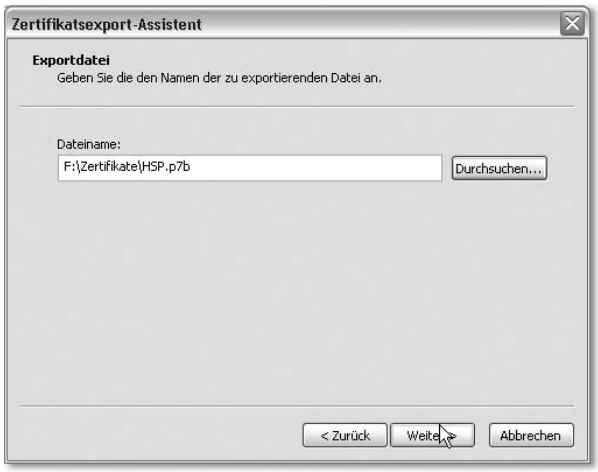

Abbildung 9.16 Auswählen der Exportdatei

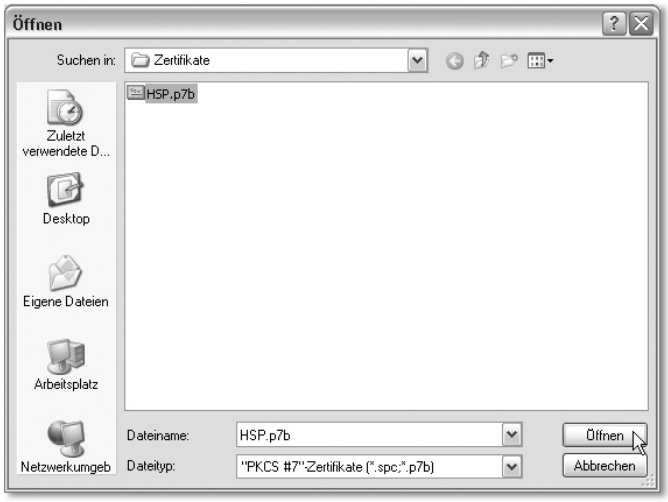

Abbildung 9.17 Auswählen des zu importierenden Zertifikats

9. Klicken Sie in den nächsten beiden Dialogen auf **Weiter** und zum Schluss auf **Fertig stellen**. Sie müssen dann noch bestätigen, dass das Zertifikat hinzugefügt werden soll, indem Sie den Dialog mit **Ja** bestätigen (siehe Abbildung 9.18).
10. Schließen Sie nun die beiden Dialoge des Internet Explorers mit **Schließen** und **OK**.

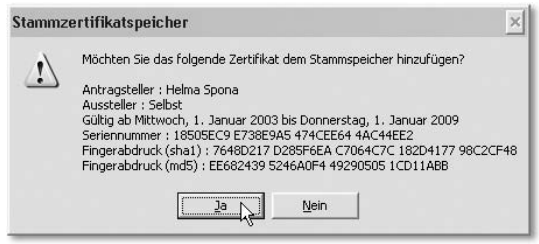

Abbildung 9.18 In diesem Dialog müssen Sie das Hinzufügen des Zertifikats bestätigen.

11. Wenn Sie nun das signierte Skript ausführen, wird die Signatur korrekt angezeigt und der Benutzer kann aufgrund der Signatur selbst entscheiden, wie er verfahren möchte. Durch Aktivierung des Kontrollkästchens **Inhalt von ... immer vertrauen** können Sie außerdem bestimmen, dass die Sicherheitswarnung in Zukunft nicht mehr erscheint, wenn das Skript mit dem gleichen Zertifikat signiert ist.

Abbildung 9.19 Anzeige bei Verwendung eines Zertifikats aus einer vertrauenswürdigen Quelle

9.3.4 Die Sicherheit des Windows Script Host aktivieren

Wenn Sie einstellen möchten, ob unsignierte Skripte, Skripte mit fehlerhafter Signatur oder nur Skripte mit gültiger Signatur ausgeführt werden sollen, gibt es dazu zwei Möglichkeiten. Befindet sich in Ihrem Netzwerk ein Domänenserver mit Windows 2000 oder Windows 2003, können Sie dies über die Sicherheitsrichtlinien einstellen. Für Netzwerke ohne Windows-2000-/2003-Server können Sie für jeden Rechner die Einstellungen separat festlegen. Das funktioniert natürlich auch, wenn Sie Ihren Rechner ganz ohne Netzwerk betreiben. Dazu müssen Sie aber manuell oder per Skript einen Registry-Eintrag setzen. Der Schlüssel `\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows`

`Script Host\Settings\TrustPolicy`, der allerdings nicht standardmäßig vorhanden ist, bestimmt die Sicherheitseinstellungen. Als Werte kommen die Werte aus Tabelle 9.1 in Frage. Wichtig ist allerdings, dass Sie den Wert als REG_DWORD-Wert schreiben.

Wert	Bedeutung
0	Führt alle Skripte aus.
1	Fragt nach, wenn ein unsigniertes Skript ausgeführt werden soll.
2	Führt nur signierte Skripte aus.

Tabelle 9.1 Mögliche Registry-Werte für die Sicherheitseinstellungen des WSH

Unter Windows XP müssen Sie mit einem weiteren Schlüssel sicherstellen, dass die Sicherheitseinstellungen auch aktiv sind. Sie müssen dazu den Schlüssel `HKLM\SOFTWARE\Microsoft\Windows Script Host\Settings\UseWINSAFER` auf 0 setzen. Die folgende HTA-Datei hilft Ihnen dabei, den Registry-Wert zu setzen und die aktuelle Einstellung auszulesen. Nach Anzeige des Formulars sorgt die Prozedur `aktuelleEinstellung` dafür, dass die aktuelle Einstellung ausgelesen und der entsprechende Listeneintrag ausgewählt wird. Ein Klick auf die Schaltfläche führt die Prozedur `Wertsetzen` aus, die die entsprechenden Änderungen in der Registry vornimmt.

```
<html>
<head>
<hta:application id="oHTA"
 applicationname="WSH-Sicherheit"
...
 sysmenu="yes">
<title>WSH-Sicherheit</title>
<script type="text/vbscript">
   Dim objWSHShell
   Dim aktWert
   set objWSHShell=CreateObject("WScript.Shell")
   Sub starten(strPfad)
      Dim objWSHShell
      Set objWSHShell=CreateObject("WScript.Shell")
      objWSHShell.Run strPfad
      Set objWSHShell=nothing
   End Sub

   Sub Wertsetzen()
```

```
      objWSHShell.RegWrite _
         "HKLM\SOFTWARE\Microsoft\Windows Script Host" & _
         "\Settings\TrustPolicy", _
         document.forms(0).stufe.selectedIndex, _
         "REG_DWORD"
      objWSHShell.RegWrite _
         "HKCU\SOFTWARE\Microsoft\Windows Script Host" & _
         "\Settings\TrustPolicy", _
         document.forms(0).stufe.selectedIndex, _
         "REG_DWORD"
      objWSHShell.RegWrite _
         "HKLM\SOFTWARE\Microsoft\Windows Script Host\" & _
         "Settings\UseWINSAFER","0"
   End Sub

   Sub aktuelleEinstellung()
      On Error Resume Next
      aktWert=objWSHShell.RegRead( _
         "HKLM\SOFTWARE\Microsoft\Windows Script Host" & _
         "\Settings\TrustPolicy")
      If Err.Number <>0 Then
         aktWert=0
         Err.Clear
      End If
      document.forms(0).stufe.selectedIndex=aktWert
   End Sub
</script>
</head>
<body onload="aktuelleEinstellung">
<form>
   <label>Aktuelle Sicherheitseinstellung für den WSH:
      <select id="stufe">
         <option value="0">
            Alle Skripte werden ausgeführt
         </option>
         <option value="1">
            Der Benutzer entscheidet, ob
            unsignierte Skripte ausgeführt
            werden sollen
         </option>
         <option value="2">Nur signierte Skripte
```

```
            werden ausgeführt
         </option>
      </select>
      <hr>
      <input type="button" onclick="Wertsetzen"
         value="Einstellung ändern">
   </label>
</form>
</body>
</html>
```

Hinweis Die HTA-Datei legt die Einstellungen für den ganzen Rechner im Hauptschlüssel `HKEY_LOCAL_MACHINE` und gesondert für den aktuellen Benutzer fest, so dass nach Festlegen der Einstellung auch andere Benutzer des Rechners den gleichen Einschränkungen unterliegen.

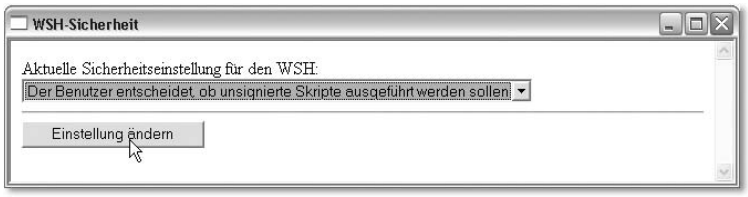

Abbildung 9.20 Die HTA-Datei zeigt die aktuelle Einstellung an und ermöglicht deren Änderung.

Wenn Sie die in Abbildung 9.20 gezeigte Einstellung nutzen, werden Sie bei jedem ausgeführten Skript gefragt, ob es ausgeführt werden soll, es sei denn, es ist signiert.

Abbildung 9.21 Mit dieser Meldung wird der Benutzer nun konfrontiert, wenn ein nicht signiertes Skript ausgeführt werden soll.

Index

F

G

H

I

P

Q

R

S

T

U

V

W

X

Z