Windows Scripting lernen

Von Windows Script Host und Visual Basic Script bis zur Windows PowerShell

Bearbeitet von
www.IT-Visions.de

1. Auflage 2016. Buch. 554 S.
ISBN 978 3 446 44800 1
Format (B x L): 18 x 24,5 cm
Gewicht: 1145 g

HANSER

Leseprobe

Holger Schwichtenberg

Windows Scripting lernen

Von Windows Script Host und Visual Basic Script bis zur Windows PowerShell

ISBN (Buch): 978-3-446-44800-1

ISBN (E-Book): 978-3-446-44944-2

Weitere Informationen oder Bestellungen unter

http://www.hanser-fachbuch.de/978-3-446-44800-1

sowie im Buchhandel.

Inhalt

Vorwort

Vorwort zur sechsten Auflage (2016)

Liebe Leserinnen, liebe Leser,

mittlerweile gibt es Windows 10 und Windows Server 2016. Aber den Verlag Addison-Wesley, bei dem dieses Buch zehn Jahre lang in fünf Auflagen erschienen ist, gibt es nicht mehr. Nun hat das Buch im Carl Hanser Verlag eine schöne neue Heimat gefunden.

Neu in dieser Auflage ist die Version 5.0 der Windows PowerShell. Zur PowerShell gibt es viele neue Skripting-Beispiele. Ebenso wurde das Buch auf Windows 10 und Windows Server 2016 aktualisiert. Bewusst sind aber nicht alle Bildschirmabbildungen mit diesen neuesten Betriebssystemen gemacht, da das Buch sich weiterhin als von der Betriebssystemversion unabhängiges Werk versteht, das auch Leser anspricht, die nicht die neueste Betriebssystemversion verwenden können oder wollen.

Viel Erfolg mit diesem Buch wünscht Ihnen

Dr. Holger Schwichtenberg

Essen, im Januar 2016

Vorwort zur fünften Auflage (2012)

Liebe Leserinnen, liebe Leser,

das Erscheinen von Windows 8 und Windows Server 2012 nehmen wir zum Anlass für eine erneute Aktualisierung dieses Buchs. Auch hier ist der Windows Script Host weiterhin enthalten und ein wichtiges Werkzeug für die automatisierte Systemadministration.

„Windows Scripting Lernen“ ist das letzte verbliebene Buch zum Windows Scripting Host (WSH) auf dem deutschen Markt.

Neben einigen Aktualisierungen zum WSH (insbesondere hinsichtlich der restriktiveren Vergabe administrativer Rechte in Windows 8) bietet diese 5. Auflage Ihnen vor allem mehr Inhalte zum Thema Windows PowerShell. Behandelt wird die PowerShell 3.0, die in Windows 8 und Windows Server 2012 enthalten ist und auf Windows 7 und Windows Server 2008 (inkl. R2) als Zusatz installierbar ist.

Ich danke Ihnen für Ihre Treue zu diesem Buch.

Dr. Holger Schwichtenberg

Essen, im September 2012

Vorwort zur vierten Auflage (2009)

Liebe Leserinnen, liebe Leser,

das Erscheinen von Windows 7 und Windows Server 2008 R2 nehmen wir zum Anlass für eine Aktualisierung des Buchs. Die weiterhin hohen Verkaufszahlen im Zeitalter von Vista und PowerShell zeigen, dass der Windows Scripting Host (WSH) in den Unternehmen noch aktiv genutzt wird, selbst wenn er in den letzten Jahren keine großen Veränderungen mehr erfahren hat.

Natürlich ist der Markt kleiner geworden. In Deutschland gab es einmal sechs Bücher zum WSH. Von diesen sind nur noch „Windows Scripting Lernen“ und der große Bruder „Windows Scripting“ übrig geblieben.

Mittelfristig wird die PowerShell größere Bedeutung als der WSH erlangen. In diesem Buch gebe ich Ihnen einen Ausblick auf die PowerShell 2.0. Aber dieser Ausblick ist hier bewusst kurz gewählt. In meinen Büchern „Windows Scripting (6. Auflage)“ und „Windows PowerShell 2.0 – Das Praxishandbuch“ (beide bei Addison-Wesley erschienen) gehe ich genauer auf die PowerShell ein.

Ich danke Ihnen für Ihre Treue zu diesem Buch.

Dr. Holger Schwichtenberg

Essen, im Oktober 2009

Vorwort zur dritten Auflage (2007)

Liebe Leserinnen, liebe Leser,

auch mit Erscheinen von Windows Vista und der Windows PowerShell ist der Windows Script Host (WSH) noch aktuell, und er wird es auch für die nächsten Jahre noch bleiben. Windows Vista basiert entgegen früheren Ankündigungen noch nicht auf dem .NET Framework, sondern weiterhin komplett auf dem Component Object Model (COM) und noch älteren C/C++-Techniken. Der WSH und seine COM-basierten Scripting-Komponenten sind also das primäre Instrument für die automatisierte Systemadministration unter Vista.

Die Windows PowerShell, die Microsoft als gemeinsamen Nachfolger von WSH und Windows-Kommandozeilen-Shell ansieht, ist zwar mächtig und einfach, steht aber hinsichtlich des direkt nutzbaren Funktionsumfangs (in Form der Commandlets) noch weit hinter dem WSH zurück. Noch muss man hier in vielen Fällen in die Tiefen von .NET einsteigen. Erst mit kommenden Windows-Versionen und anderen Microsoft-Produkten wird es für die PowerShell einen Funktionsumfang geben, der auch Scripting-Einsteiger anspricht.

Grund genug also, diesem meistverkauften deutschen Scripting-Buch eine neue Auflage zu spendieren. In dieser Neuauflage finden Sie neben vielen kleinen Verbesserungen neue Texte zu folgenden Themen:

- Scripting-Neuerungen in Windows Vista (Kapitel 19)
- Einführung in die Windows PowerShell (Kapitel 20)
- Prozessverwaltung per Skript, insbesondere die Kommunikation zwischen Skripten und Konsolenanwendungen (Kapitel 16)

- Mehr Beispiele zur Verwendung von Text- und XML-Dateien als Ein- und Ausgabedateiformat für Skripte (in mehreren Kapiteln)

Beim Alten geblieben ist die Website, auf der Sie sich registrieren können für die Foren, Zusatz-Downloads und den Newsletter:

http://www.Windows-Scripting.de

Weiterhin viel Spaß beim Skripten wünscht Ihnen

Dr. Holger Schwichtenberg

Essen, im Mai 2007

Vorwort zur zweiten Auflage (2004)

Dass viele Administratoren sich ein Einsteigerbuch zum WSH wünschten, war mir klar, als ich dieses Buch Ende 2002 zusammen mit meinen drei Co-Autoren geschrieben habe. Dass wir damit die Position des Marktführers unter den Scripting-Büchern in Deutschland einnehmen würden, hätte ich nicht erwartet. Natürlich freuen wir uns sehr über die positive Resonanz.

Bereits Ende 2003 ist ein korrigierter Nachdruck erschienen, in dem wir die restlichen kleinen Tippfehler in dem Buch und auf der CD beseitigt haben. Nun liegt eine überarbeitete und erweiterte zweite Auflage vor Ihnen. Komplett neu in diesem Buch sind die Kapitel 16 („Scripting der Gruppenrichtlinien“) und 17 („Sicheres Scripting“).

Herzlich bedanken möchte ich mich bei allen Lesern, die durch ihr Feedback geholfen haben, diese zweite Auflage noch besser zu machen.

Ausdrücklich hinweisen möchte ich Sie auf die Website zu diesem Buch:

http://www.Windows-Scripting.de

Aktuell bietet Ihnen diese Website folgende Informationen und Dienste:

- Umfangreiches Windows Scripting-Glossar
- FAQ zum WSH
- Diskussionsforum zum Windows Scripting (Fragen von registrierten Lesern werden von den Autoren dieses Buchs vorrangig beantwortet!)
- Verzeichnis von Scripting-Komponenten
- Klassenreferenz für die Windows Management Instrumentation (WMI)
- Feedback-Fragebogen zu diesem Buch
- Aktualisierungen zu den Skripten in diesem Buch (sofern sich technische Änderungen in Windows ergeben oder Verbesserungen von uns oder den Lesern gefunden werden)
- Skriptarchiv mit über 200 weiteren WSH-Skripten
- Scripting-News und -Newsletter
- Und last, but not least: Informationen zu unseren Scripting-Schulungen und zum Support bei Fragen rund um den WSH.

Ich wünsche Ihnen nun viel Erfolg mit diesem Buch und würde mich freuen, Sie auf meiner Website begrüßen zu dürfen!

Dr. Holger Schwichtenberg

Essen, im Mai 2004

Vorwort zur ersten Auflage (2002)

WSH

Zur automatisierten Systemadministration ist der Windows Script Host (WSH) eine sehr mächtige Alternative gegenüber der schon etwas angestaubten Windows-Batch-Programmierung. Unsere Erfahrungen aus Scripting-Schulungen und Beratungsterminen in den letzten vier Jahren haben aber gezeigt, dass es vielen Administratoren schwerfällt, sich in die Welt des Scriptings einzuarbeiten – oft auch gehemmt durch die Tatsache, dass das Scripting zum Bereich Programmierung/Softwareentwicklung gezählt wird.

Zielgruppe

Scripting ohne Vorkenntnisse

„Windows Scripting Lernen“ wendet sich an Administratoren ohne Programmierkenntnisse. Dieses Buch enthält eine schrittweise Einführung in die Entwicklung von Skripten. Auch ohne Vorerfahrung in der Programmierung lernen Sie durch dieses Buch die Möglichkeiten zur automatisierten Administration von Unternehmensnetzwerken mit dem Windows Script Host (WSH), Visual Basic Script und verschiedenen sogenannten COM-Komponenten kennen.

Methodik

Didaktischer Aufbau

Das Buch hat eine didaktische Struktur mit aufeinander aufbauenden Kapiteln. Bewusst wird darauf verzichtet, detaillierte Hintergründe sowie jede Möglichkeit und jede Option vorzustellen. Dieses Buch fokussiert auf das Wesentliche, um Ihnen einen leichten Einstieg in das Windows Scripting zu ermöglichen.

Einführung

Alle grundlegenden Konzepte der Programmierung wie Variablen, Fallunterscheidungen, Schleifen, Fehlerbehandlung und die Arbeit mit Komponenten, Klassen und Objekten werden von Grund auf eingeführt. Außerdem finden Sie eine ausführliche Erklärung zur Installation und Konfiguration der Skripte und Komponenten sowie Hinweise auf mögliche Probleme oder Fehlersituationen.

Die Beispiele sind bewusst einfach gehalten. Dennoch werden Sie lernen, alle wesentlichen Aufgaben der System- und Netzwerkadministration durch Skripte zu lösen. Der deutliche Schwerpunkt dieses Buches liegt nicht auf dem Scripting im Heimeinsatz, sondern auf dem Scripting in Unternehmensnetzwerken. Daher finden Sie hier auch Themen wie das Scripting des Active Directory, der Netzwerkkonfiguration und von Ereignisprotokollen.

Am Ende jedes Kapitels stehen Aufgaben, die Sie einsetzen können, um Ihr Wissen zu vertiefen und praktisch zu üben. Gewisse Wiederholungen sind in diesem Einsteigerbuch übrigens kein Fehler, sondern didaktische Absicht.

Wie Sie dieses Buch lesen sollten

Leseanleitung

Aufgrund des didaktischen Konzepts sollten Sie die ersten fünf Kapitel dieses Buches unbedingt sequenziell in der vorgegebenen Reihenfolge lesen. Ab Kapitel 6 werden dann verschiedene Gebiete des Scriptings aufgabenorientiert behandelt. Die Kapitel 6 bis 16 müssen Sie nicht notwendigerweise sequenziell lesen. Hier können Sie durchaus direkt zu den Kapiteln springen, die Sie besonders interessieren. Die von uns gewählte Reihenfolge beinhaltet aber eine Steigerung im Schwierigkeitsgrad.

Am Ende eines jeden Kapitels gibt es einen Aufgabenteil; die passenden Lösungen stehen zusammenhängend im Anhang, sodass das „Spicken“ etwas erschwert wird.

Weitere Unterstützung im WWW

Website

Als Leser dieses Buches haben Sie Zugriff auf einen zugangsbeschränkten Bereich der deutschen Windows Scripting-Website, die Sie unter *http://www.Windows-Scripting.de* finden. Ein Service dieser Website ist, dass Sie den Autoren dieses Buches verbliebene Fragen stellen können.

Der große Bruder „Windows Scripting“

Weiterführende Literatur

Im Buchhandel werden Sie einen „großen Bruder“ zu diesem Buch finden, der schon zwei Jahre länger auf dem Markt ist: „Windows Scripting“ geht parallel zu diesem „Windows Scripting Lernen“ bei Addison-Wesley in die dritte Auflage. Dieser Titel aus der WinTec-Reihe ist ein umfassendes Nachschlagewerk zu allen Bereichen der Skriptprogrammierung und richtet sich an Entwickler und Administratoren, die bereits Vorkenntnisse in mindestens einer Programmiersprache besitzen. Wenn Sie nach der Lektüre von „Windows Scripting Lernen“ noch Wissensdurst verspüren, dann sollten Sie zum großen Bruder greifen.

Dank

Mein herzlicher Dank gilt

- meinen Co-Autoren Sven Conrad, Thomas Gartner und Oliver Scheer, die tatkräftig mitgeholfen haben, den umfangreichen Stoff aus dem „Windows Scripting“-Buch auf die „Lernen“-Reihe herunterzubrechen,
- Ayşe Aruca und Georg Meindl für ihre kritischen Anmerkungen als Testleser dieses Buches,
- meiner Korrektorin Astrid Schürmann, die wieder einmal mit hoher Präzision nicht nur die sprachlichen Fehler aus unserem Text entfernt, sondern uns auch auf inhaltliche Inkonsistenzen hingewiesen hat,
- und meiner Lektorin Sylvia Hasselbach für ihre Geduld bei der doch langwierigen Geburt dieses Werks.

Ich wünsche Ihnen nun viel Erfolg mit diesem Buch.

Holger Schwichtenberg

Essen, im November 2002

Website für Leser

Zu diesem Buch gibt es eine eigene Website:

http://www.windows-scripting.de

Sie als Leser haben neben den öffentlichen Bereichen auch die Möglichkeit, auf einen geschützten Bereich zuzugreifen, der besondere Informationen enthält:

Downloads: die aktuellen Versionen der in diesem Buch abgedruckten Skripte sowie weitere Skripte und Codebeispiele

Verzeichnis	Inhalt
\Skripte	Alle Skripte aus dem Buch, geordnet nach Kapiteln
\Install	Erweiterungen, Komponenten, Sprachen und Tools für das Windows Scripting (zum Teil als Vollversionen, zum Teil als Demoversionen)
\Weitere Informationen	Dieses Verzeichnis enthält zusätzliche Dokumentationen zu Visual Basic Script, dem WSH und einigen der besprochenen Komponenten.
\Über den Autor	Informationen über den Autor dieses Buchs

Foren: Wenn Sie Fragen haben oder Ihre Meinung zu einem Thema dieses Buchs äußern möchten, dann können Sie hier auf Reaktionen anderer Nutzer hoffen.

Leser-Bewertung: Vergeben Sie Noten für dieses Buch und lesen Sie nach, was andere Leser von diesem Buch halten.

Bug-Report: Melden Sie hier Fehler, die Sie in diesem Buch gefunden haben! Hier können Sie auch nachlesen, welche Fehler anderen nach Drucklegung aufgefallen sind.

Newsletter: Alle registrierten Leser erhalten in unregelmäßigen Abständen einen Newsletter.

Der URL für den Zugang zum Leser-Portal lautet:

http://www.windows-scripting.de/leser

Bei der Anmeldung müssen Sie das Erstzugangskennwort Defiance Skies angeben (Defiance ist eine Science-Fiction-Serie).

Bitte beachten Sie, dass das Leser-Portal eine freiwillige, private Leistung des Autors ist, auf die es keinen Rechtsanspruch gibt.

Über den Autor
Dr. Holger Schwichtenberg

www.IT-Visions.de®
Dr. Holger Schwichtenberg

5Minds
IT-SOLUTIONS

- Studienabschluss Diplom-Wirtschaftsinformatik an der Universität Essen
- Promotion an der Universität Essen im Gebiet komponentenbasierter Softwareentwicklung
- Seit 1996 selbstständig als unabhängiger Berater, Dozent, Softwarearchitekt und Fachjournalist
- Leiter des Berater- und Dozententeams bei *www.IT-Visions.de*
- Leitung der Softwareentwicklung im Bereich Microsoft/.NET bei der 5minds IT-Solutions GmbH & Co. KG *(www.5minds.de)*
- Lehrbeauftragter an den Fachhochschulen in Münster und Graz
- 60 Fachbücher bei zahlreichen Verlagen und mehr als 800 Beiträge in Fachzeitschriften
- Gutachter in den Wettbewerbsverfahren der EU gegen Microsoft (2006 – 2009)
- Ständiger Mitarbeiter der Zeitschriften iX (seit 1999), dotnetpro (seit 2000) und Windows Developer (seit 2010) sowie beim Online-Portal heise.de (seit 2008)
- Regelmäßiger Sprecher auf nationalen und internationalen Fachkonferenzen (z. B. Microsoft TechEd, Microsoft Summit, Microsoft IT Forum, BASTA, BASTA-on-Tour, .NET Architecture Camp, Advanced Developers Conference, Developer Week, OOP, DOTNET Cologne, VS One, NRW.Conf, Net.Object Days, Windows Forum)

- Zertifikate und Auszeichnungen von Microsoft:
  - Microsoft Most Valuable Professional (MVP)
  - Microsoft Certified Solution Developer (MCSD)
- Thematische Schwerpunkte:
  - Microsoft .NET Framework, Visual Studio, C#, Visual Basic
  - .NET-Architektur/Auswahl von .NET-Technologien
  - Einführung von .NET Framework und Visual Studio/Migration auf .NET
  - Webanwendungsentwicklung mit IIS, ASP.NET und JavaScript sowie TypeScript
  - Verteilte Systeme, Webservices, Enterprise .NET
  - Relationale Datenbanken, XML, Datenzugriffsstrategien
  - Objektrelationales Mapping (ORM), insbesondere ADO.NET Entity Framework
  - Windows PowerShell (WPS) und Windows Management Instrumentation (WMI)
- Ehrenamtliche Community-Tätigkeiten:
  - Vortragender für die International .NET Association (INETA)
  - Betrieb diverser Community-Websites *www.dotnetframework.de*, *www.entwickler-lexikon.de*, *www.windows-scripting.de*, *www.aspnetdev.de* u. a.
- Weblog: *www.dotnet-doktor.de*
- Kontakt: *buero@IT-Visions.de* sowie Telefon 02 01 64 95 90-0

9 Scripting der Benutzerverwaltung

Lernziele

Dieses Kapitel versetzt den Administrator in die Lage, selbst komplexere Vorgänge in der Benutzerverwaltung durch das Zusammenführen einzelner Vorgänge zu vereinfachen. Benutzerverwaltung soll hier im weiteren Sinne auch Benutzergruppen und Benutzercontainer umfassen.

Vereinfachte Administration

Die Verwaltung von Benutzerkonten in Unternehmensnetzen gewinnt immer mehr an Bedeutung. Während das Verwalten einzelner Benutzer durch den Administrator noch in endlicher Zeit erledigt werden kann, gestaltet sich das Verwalten der Benutzerkonten in komplexen Netzwerken sehr aufwendig. Hier verschafft die Skriptprogrammierung dem Administrator die Möglichkeit, lästige Aufgaben durch einfaches Aufrufen von Skripten zu erledigen.

HINWEIS: Aufgrund unterschiedlicher Anforderungen und Vorgehensweisen ist dieses Kapitel getrennt in die Benutzerverwaltung für lokale Benutzerkonten und Active-Directory-basierte Systeme. Die Benutzerverwaltung für lokale Benutzer in Windows-Clients und Windows Server erfolgt heute in allen Windows-Betriebssystemen bis hin in Windows 10 und Windows Server 2016 noch auf die gleiche Weise wie einst in Windows NT. Die Active-Directory-Benutzerverwaltung kann wirklich nur auf das Active Directory angewendet werden.

ADSI

ADSI

Die Benutzerverwaltung basiert auf der Komponente ADSI. Es gibt zwar auch einige Klassen in WMI für die Benutzerverwaltung, die Verwaltung mit ADSI ist jedoch einfacher und vollständiger, sodass sie hier verwendet wird.

HINWEIS: Als wichtige Begriffe seien noch einmal wiederholt: Ein Container ist ein Verzeichnisobjekt, das andere Verzeichnisobjekte enthalten darf. Über einen Container kann man mit `For Each` eine Schleife bilden. Ein Blatt ist ein Verzeichnisobjekt, das keine Unterobjekte enthält; somit ist eine `For Each`-Schleife nicht möglich.

9.1 Benutzerverwaltung für lokale Benutzerkonten

Flache Strukturen

Die Frage, ob nicht Active-Directory-basierte Windows-Versionen überhaupt einen Verzeichnisdienst haben, führt gewöhnlich zu hitzigen Diskussionen, da diese Betriebssysteme alle Verzeichnisobjekte in flachen Strukturen verwalten. Es existieren nur einige wenige Container und auch das Anlegen von eigenen Untercontainern wird nicht unterstützt. Aus Gründen der Einfachheit verwenden wir hier jedoch den Begriff Verzeichnisdienst auch für NT4-Domänen.

HINWEIS: Ebenfalls aus Gründen der Vereinfachung wird in diesem Kapitel immer von der NT4-Benutzerverwaltung gesprochen.

Die hier vorgestellten Verfahren gelten für:

- Windows-NT-Domänen (vor Windows 2000),
- Windows-Client ab Version 2000 Professional bis zum heute aktuellen Windows 10,
- Windows Server ab Version 2003 (bis einschließlich des heute aktuellen Windows Server 2016), die nicht Domänencontroller sind.

9.1.1 Anlegen eines Benutzerkontos

Benutzer erzeugen

Vor dem Anlegen eines neuen NT-Benutzerkontos muss zunächst die Bindung an den übergeordneten `Domain`- oder an einen `Computer`-Container hergestellt werden. Dazu wird bei `GetObject()` ein ADSI-Pfad zu einem Computer oder einer Domäne angegeben. Der Pfad ist sehr einfach:

`WinNT://COMPUTERNAME` oder `WinNT://DOMÄNENNAME`

ACHTUNG: Auch wenn dies in Kapitel 5 schon mehrfach erwähnt wurde, sei hier dennoch erneut die Warnung ausgesprochen: WinNT müssen Sie mit großen (W), (N), (T) und kleinem (i), (n) schreiben. Die häufigste Ursache für nicht funktionierende ADSI-Skripte ist die falsche Schreibweise dieses Begriffs. Dieser Fehler tritt so häufig auf, weil die Relevanz der Groß- und Kleinschreibung in der VBScript-Programmierung sehr selten ist.

TIPP: Die zusätzliche Angabe des Klassennamens im ADSI-Pfad beschleunigt den Aufruf, weil ADSI dann genau weiß, wonach es suchen soll. Der Klassenname kann am Ende des Pfads durch ein Komma getrennt angegeben werden:

`WINNT://COMPUTERNAME, Computer` oder `WinNT://DOMÄNENNAME, Domain`

Grundsätzlich wird in ADSI ein Objekt mit der Methode `Create()` angelegt. Bei der Methode `Create()` sind der Klassenname `user` und als zweiter Parameter der gewünschte Benutzername anzugeben. Erst mit dem Aufruf von `SetInfo()` wird der Benutzer tatsächlich angelegt. Create()

Die `User`-Klasse verlangt keine Pflichtattribute; im Skript werden allerdings die folgenden optionalen Attribute verwendet: Attribute

- `FullName`: kennzeichnet den Anzeigenamen des Benutzers
- `Description`: eine Beschreibung des Benutzers
- `HomeDirectory`: der Pfad zu dem Verzeichnis, in dem der Benutzer seine Daten ablegt
- `AccountExpirationDate`: Datum, an dem das Konto ungültig wird
- `PasswordExpirationDate`: Datum, an dem das Kennwort des Kontos abläuft. `PasswordExpirationDate` kann aber nicht beschrieben werden. Das Ablaufdatum kann nur beeinflusst werden über `MaxPasswordAge` auf Domänen- bzw. Computerebene. Damit der Benutzer nach dem Anmelden sein Kennwort ändern muss, setzt man `Benutzer. PasswordExpired = 1`.
- `LoginScript`: das bei der Anmeldung des Benutzers auszuführende Skript

TIPP: Tipp: Bitte passen Sie in diesem Skript unbedingt den Namen des Containers an, bevor Sie es testen. Tragen Sie in die Konstante `CONTAINER NAME` den Namen eines Computers oder einer Domäne ein, die bei Ihnen erreichbar ist. Selbstverständlich müssen Sie Administratorrechte auf dem Computer bzw. der Domäne besitzen, um das Skript ausführen zu können.

Listing 9.1: /Skripte/Kapitel08/WinNT/BenutzerAnlegen.vbs

```
' BenutzerAnlegen.vbs
' Anlegen eines Benutzerkontos
' verwendet: ADSI
' ==============================
Option Explicit
' Variablen deklarieren
Dim Benutzer
Dim Domaene
' Name des Containers, in dem der Benutzer angelegt werden soll
Const CONTAINERNAME = "PC171" ' Computername oder Domänenname
Const KLASSE = "Computer" ' oder: Domain
' Zugriff auf Domain-Objekt
Set Domaene = GetObject("WinNT://" & CONTAINERNAME & "," & KLASSE)
' Benutzer anlegen
Set Benutzer= Domaene.Create("user", "WilliWinzig3")
' Setzen von Eigenschaften
' Voller Name
Benutzer.FullName = "Willi Winzig"
' Beschreibung des Benutzers
Benutzer.Description = "Herr Willi Winzig ist unser neuer Mitarbeiter."
' Home-Verzeichnis des Benutzers
Benutzer.HomeDirectory = "e:\homes\winzig"
' Ablaufdatum des Kontos: 1 Jahr
```

```
Benutzer.AccountExpirationDate = Now() + 365
' Verweis auf das Login-Skript
Benutzer.LoginScript = "benutzer.bat"
' Kennwort setzen
Benutzer.SetPassword "SehrGeheim123"
' Kennwortänderung bei erster Anmeldung erzwingen
Benutzer.PasswordExpired = 1
' Festschreiben der Werte
Benutzer.SetInfo
' Meldung ausgeben
WScript.Echo "Benutzer angelegt!"
```

In den folgenden Bildschirmabbildungen werden bewusst verschiedene neuere Betriebssysteme verwendet, um zu beweisen, dass diese Vorgehensweise auch in modernen Betriebssystemen und im Zeitalter des Active Directory noch relevant ist. Viele Administratoren glauben fälschlicherweise, die Benutzerkontenverwaltung in einem Netzwerk mit Active Directory würde komplett über LDAP laufen.

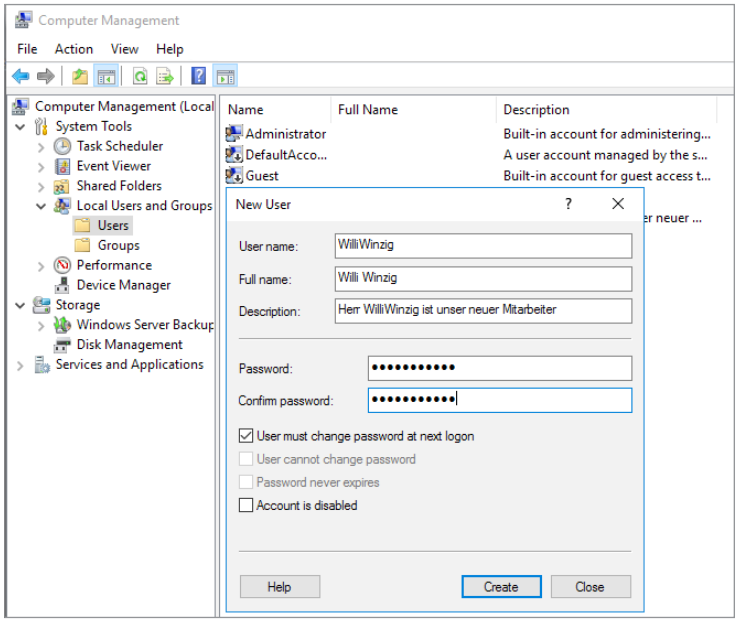

Abbildung 9.1: Anlegen des Benutzers Willi Winzig als lokalen Benutzer

Der neue Benutzer WilliWinzig erscheint aber nicht in der Benutzerkontenverwaltung der Systemsteuerung, weil er nicht Mitglied der Standardgruppe „Benutzer“ ist. Der neu angelegte Benutzer gehört zunächst zu keiner Gruppe und er hat auch noch kein Kennwort. Diese beiden Schritte erfolgen in den nächsten Skripten.

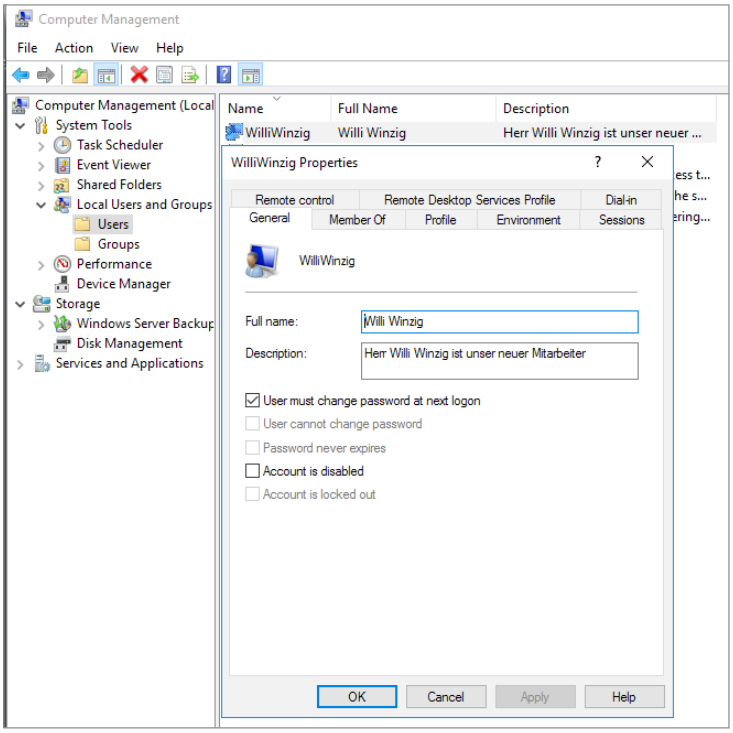

Abbildung 9.2: Anzeige des neuen Benutzerkontos (hier: Windows Server 2016)

9.1.2 Umbenennen eines Benutzers

Namens-änderung

Der NT4-Verzeichnisdienst erlaubt die Umbenennung eines Benutzerkontos nach dem Anlegen, da für die eindeutige Identifizierung nicht der Kontoname, sondern der Security Identifier (SID) des Kontos maßgeblich ist. Das Konto verliert also nicht seine Gruppenzuordnungen oder Rechte. Die Methode zur Umbenennung heißt in ADSI `MoveHere()`. Diese Methode wird sowohl von der `Computer`- als auch von der `Domain`-Klasse unterstützt.

ACHTUNG: Es ist nicht möglich, ein lokales Benutzerkonto zu verschieben, weil es pro Computer nur einen Container für Benutzer geben kann. Eine Verschiebung zwischen Containern ist nicht möglich.

Das Skript deklariert die benötigten Variablen für die Objekte. Durch den Aufruf von `GetObject()` wird eine Instanz des `Domain`-Objekts erzeugt und der Variablen `Container` zugewiesen. Als Parameter werden der WinNT-Provider und der Name des Computers angegeben, auf dem sich das Benutzerkonto befindet.

Umbenennen durch Verschieben

Der Aufruf der `MoveHere()`-Methode des `Domain`-Objekts mit dem ADSI-Pfad des Benutzers sowie dem neuen Benutzernamen führt die Umbenennung durch. Die erfolgreiche Umbenennung wird durch eine Meldung angezeigt.

Listing 9.2: /Skripte/Kapitel08/WinNT/BenutzerUmbenennen.vbs

```
' BenutzerUmbenennen.vbs
' Umbenennen eines Benutzerkontos
' verwendet: ADSI
' ==============================
Option Explicit

' Konstanten - bitte anpassen!!!
Const CONTAINERNAME = "PC171" ' Computername oder Domänenname
Const ALTERNAME = "WilliWinzig"
Const NEUERNAME = "WilliWichtig"

' Notwendige Variablen deklarieren
Dim Container
' Zugriff auf Domain-Objekt
Set Container = GetObject("WinNT://" & CONTAINERNAME)
' MoveHere ausführen
Container.MoveHere _
"WinNT://" & CONTAINERNAME & "/" & ALTERNAME,NEUERNAME
' Meldung ausgeben
Wscript.Echo "Benutzer umbenannt!"
```

Wie Sie in nachfolgender Bildschirmabbildung sehen können, wird durch die `MoveHere()`-Methode nur der Benutzername selbst, nicht aber die anderen Attribute wie `FullName` und `Description` beeinflusst.

ACHTUNG: Bitte beachten Sie, dass nach Ausführung dieses Skripts das Benutzerkonto „WilliWinzig“ nicht mehr existiert. Da die nachfolgenden Skripte „WilliWinzig“ verwenden, sollten Sie mithilfe des ersten Skripts in diesem Kapitel „WilliWinzig“ wieder anlegen.

9.1.3 Kennwort eines Benutzers ändern

Grundsätzlich gibt es zwei Möglichkeiten, ein Kennwort mit ADSI zu setzen:

Set Password()

- Bei `SetPassword()` ist die Angabe des bisherigen Kennworts nicht nötig.
- Bei der Methode `ChangePassword()` muss das bisherige Kennwort angegeben werden.

Change Password()

`ChangePassword()` sollte angewendet werden, wenn sichergestellt werden soll, dass nur der betreffende Benutzer selbst das Kennwort ändert. Die Methode lässt sich nur ausführen, wenn die Kontorichtlinien dies erlauben (wenn Sie das Skript ausgeführt haben, das die minimale Kennwortdauer auf zehn Tage setzt, dann kann `ChangePassword()` erst nach zehn Tagen zum ersten Mal ausgeführt werden!).

TIPP: Für den Administrator ist die Methode `SetPassword()` gedacht, da das alte Kennwort nicht bekannt sein muss. `SetPassword()` kann nicht nur beim erstmaligen Setzen, sondern zu beliebiger Zeit ausgeführt werden.

ACHTUNG: In älteren Windows-Versionen (vor Windows Server 2003 und Windows XP mit Service Pack 2) konnte das Kennwort beim Anlegen eines neuen Benutzers erst gesetzt werden, nachdem das Anlegen mit `SetInfo()` vollzogen wurde. Damit ist eine potenzielle Sicherheitslücke geschlossen.

Für den LDAP-Provider gilt jedoch die Aussage „Erst Konto komplett anlegen, dann Kennwort setzen“ immer noch. Das potenzielle Risiko kann hier dadurch umgangen werden, dass das Konto, das im Standard deaktiviert ist, erst nach der Kennwortvergabe aktiviert wird!

Listing 9.3: /Skripte/Kapitel08/WinNT/KennwortAendern1.vbs

```
' KennwortAendern1.vbs
' Setzen eines Kennworts für ein Benutzerkonto
' verwendet: ADSI
' ================================================================
Dim Benutzer
' Bitte ADSI-Pfad anpassen: WinNT://CONTAINER/BENUTZERNAME
Set Benutzer = GetObject("WinNT://PC171/WilliWinzig,user")
Benutzer.SetPassword "Helmut"
Msgbox "Kennwort gesetzt!"
```

Listing 9.4: /Skripte/Kapitel08/WinNT/KennwortAendern2.vbs

```
' KennwortAendern2.vbs
' Ändern eines Kennworts für ein Benutzerkonto
' verwendet: ADSI
' ================================================================

Dim Benutzer
' Bitte ADSI-Pfad anpassen: WinNT://CONTAINER/BENUTZERNAME
Set Benutzer = GetObject("WinNT://PC171/WilliWinzig,user")
Benutzer.ChangePassword "Helmut", "Gerhard"
Msgbox "Kennwort geändert!"
```

TIPP: Um den Benutzer zu veranlassen, sein Kennwort bei der nächsten Anmeldung zu ändern, wird die Eigenschaft `AccountExpirationDate` auf das aktuelle Datum und die aktuelle Uhrzeit gesetzt.

9.1.4 Anlegen einer Benutzergruppe

Das Einrichten einer Gruppe erfolgt analog zur Erstellung eines `User`-Objekts. Beachten Sie aber den bei `Create()` anzugebenden Klassennamen `Group`.

Andere Klasse

Lokal oder global

`GroupType` ist ein Pflichtattribut des lokalen Benutzerkontos, das aber automatisch auf den Wert 2 (globale Gruppe) gesetzt wird, wenn der ADSI-Client keinen Wert vorgibt. Das Beispielskript allerdings erzeugt eine lokale Gruppe (Wert 4).

Listing 9.5: /Skripte/Kapitel08/WinNT/GruppeAnlegen.vbs

```
' GruppeAnlegen.vbs
' Anlegen einer lokalen Gruppe
' verwendet: ADSI
' ================================================================

Option Explicit
' Variablen deklarieren
Dim Container
Dim Gruppe
' Konstanten definieren
Const GRUPPENNAME = "Finanzbeamte"
' Name des Containers, in dem der Benutzer angelegt werden soll
Const CONTAINERNAME = "PC171" ' Computername oder Domänenname

' Zugriff auf Domain-Objekt
Set Container = GetObject("WinNT://" & CONTAINERNAME)
' Erzeugen der Gruppe
Set Gruppe = Container.Create("group", GRUPPENNAME)
' Gruppentyp setzen
' 4 = Lokale Gruppe, 2= Globale Gruppe
Gruppe.Put "Grouptype", 4
' Beschreibungstext setzen
Gruppe.Description = "Beispielgruppe für das Buch 'Windows Scripting lernen'"
' Festschreiben der Änderungen
Gruppe.SetInfo
' Meldung ausgeben
WScript.Echo "Gruppe " & GRUPPENNAME & " wurde angelegt!"
```

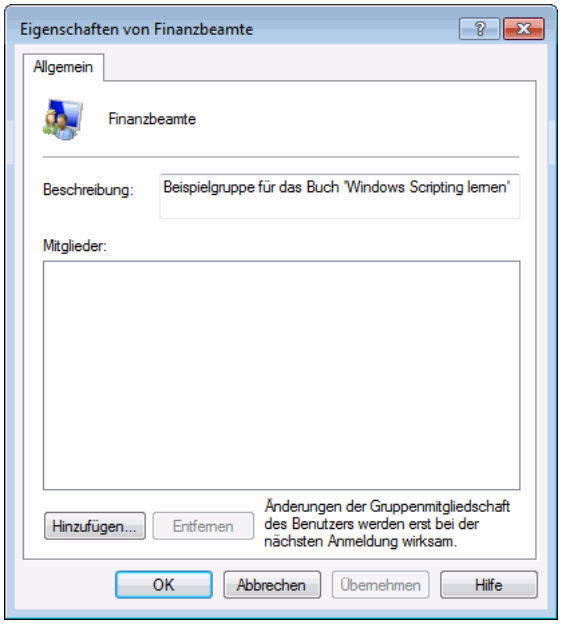

Abbildung 9.3: Neue Gruppe erstellt

9.1.5 Hinzufügen eines Benutzers zu einer Gruppe

Grup-
pieren

Das folgende Skript ordnet einen bestehenden Benutzer einer existierenden Gruppe zu. Das Skript deklariert die benötigte Variable für das `Group`-Objekt. Durch den Aufruf von `GetObject()` wird eine Instanz des `Group`-Objekts erzeugt und der Variablen `Gruppe` zugewiesen. Durch Aufruf der `Add()`-Methode des `Group`-Objekts wird der als Parameter übergebene Benutzer der Gruppe zugeordnet.

Der Befehl `SetInfo()` ist hier nicht notwendig, die Änderung wird sofort wirksam.

ACHTUNG: Der Benutzer muss bereits vorhanden sein, ansonsten wird die Fehlermeldung „Ein Mitglied konnte in der lokalen Gruppe nicht hinzugefügt oder entfernt werden, da das Mitglied nicht vorhanden ist.“ ausgegeben.

Listing 9.6: /Skripte/Kapitel08/WinNT/BenutzerzuGruppe.vbs

```
' BenutzerzuGruppe.vbs
' Hinzufügen eines Benutzers zu einer Gruppe
' verwendet: ADSI
' ================================
Option Explicit
' Variablen deklarieren
Dim Gruppe
' Zugriff auf das Gruppen-Objekt
Set Gruppe = GetObject("WinNT://PC171/Finanzbeamte,Group")
' Hinzufügen des Benutzer-Objekts zur Gruppe
Gruppe.Add ("WinNT://PC171/WilliWinzig")
' Meldung ausgeben
Wscript.Echo "Benutzer WilliWinzig zur Gruppe 'Finanzbeamte' hinzugefügt!"
```

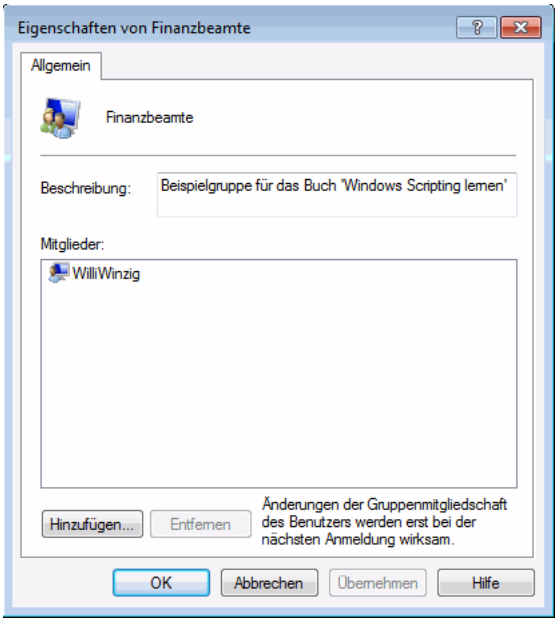

Abbildung 9.4:
Benutzer in die Gruppe eingefügt

9.1.6 Entfernen eines Benutzers aus einer Gruppe

Benutzer entfernen

Das nachfolgende Skript *LoescheBenutzerausGruppe.vbs* entfernt einen Benutzer aus einer Benutzergruppe. Zentraler Befehl ist die Methode `Remove()`, die auf einem `Group`-Objekt ausgeführt wird. Als Parameter erwartet `Remove()` den ADSI-Pfad des Benutzers, der aus der Gruppe entfernt werden soll.

Identifikation

Der Variablen `Gruppe` wird durch `GetObject()` ein Verweis auf das `Group`-Objekt der betreffenden Gruppe zugewiesen. Danach wird `Remove()` ausgeführt. Der Befehl `SetInfo()` ist hier nicht notwendig, die Änderung wird sofort wirksam.

Listing 9.7: /Skripte/Kapitel08/WinNT/LoescheBenutzerausGruppe.vbs

```
' LoescheBenutzerausGruppe.vbs
' Löschen eines Benutzerkontos aus einer Gruppe
' verwendet: ADSI
' ==============================
Option Explicit
Dim Benutzer
Dim Gruppe
' Zugriff auf das Gruppen-Objekt
Set Gruppe = GetObject("WinNT://PC171/Finanzbeamte,group")
' Benutzer-Objekt aus der Gruppe entfernen
Gruppe.Remove("WinNT://PC171/WilliWinzig,user")
Wscript.Echo "Der Benutzer WilliWinzig wurde aus der " & _
    "Gruppe Finanzbeamte entfernt."
```

9.1.7 Deaktivieren eines Benutzerkontos

Konto sperren

Soll einem Benutzer der Zugang zum Netzwerk nur kurzfristig entzogen werden, muss das Konto nicht gelöscht, sondern es kann deaktiviert werden. Ein Konto kann auch gesperrt werden. Dies hat denselben Effekt wie die Deaktivierung.

Das nachfolgende Beispiel zeigt, wie mithilfe des Attributs `AccountDisabled` ein Benutzer deaktiviert wird, sodass er sich nicht mehr am Netz anmelden kann. Dazu wird mit `GetObject()` ein Verweis auf das `User`-Objekt erstellt und in der Variablen `Benutzer` gespeichert. Durch Setzen der Eigenschaft `AccountDisabled` auf den Wert `True` wird das `User`-Objekt angewiesen, das Konto zu sperren. Die Sperrung erfolgt erst mit dem Aufruf von `SetInfo()`.

Entsperrung

Die Umkehrung der Aktion ist mit der Zuweisung des booleschen Werts `False` an die Eigenschaft `AccountDisabled` möglich.

Listing 9.8: /Skripte/Kapitel08/WinNT/DeaktiviereKonto.vbs

```
' DeaktiviereKonto.vbs
' Deaktivieren eines Benutzerkontos
' verwendet: ADSI
' ================================================================
Option Explicit
' Variablen deklarieren
Dim Benutzer
```

```
' Zugriff auf User-Objekt
Set Benutzer= GetObject("WinNT://PC171/WilliWinzig,user")
' Deaktivierung
Benutzer.AccountDisabled = True ' = False zum Reaktivieren!
' Cache schreiben
Benutzer.SetInfo
' Meldung ausgeben
Wscript.Echo "Benutzer WilliWinzig deaktiviert!"
```

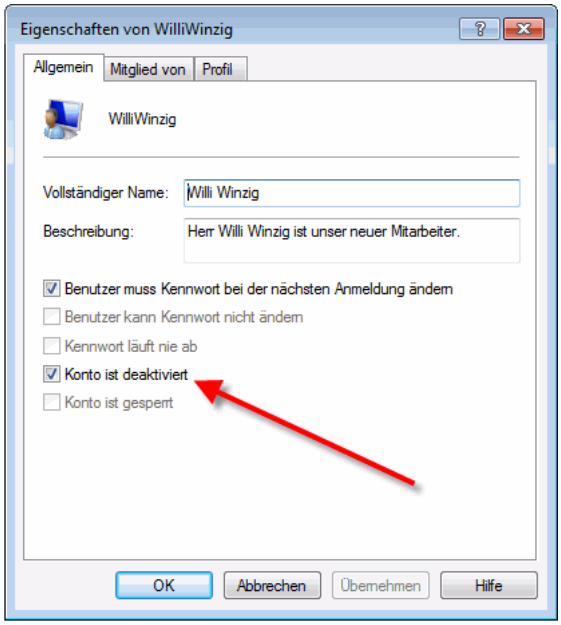

Abbildung 9.5:
Benutzerkonto ist deaktiviert.

9.1.8 Löschen einer Gruppe

Delete()

Zentraler Befehl beim Löschen eines Objekts ist die Methode `Delete()`, die nur von Container-Objekten (also den Klassen `Domain` und `Computer`) bereitgestellt wird. `Delete()` erwartet nicht nur den Namen des zu löschenden Objekts, sondern zuvor im ersten Parameter auch den Klassennamen.

Im nächsten Beispiel wird das Löschen einer Gruppe demonstriert. Dazu wird nach der Variablendeklaration der Variablen `Container` der Verweis auf das durch `GetObject()` erzeugte `Domain`- oder `Computer`-Objekt zugewiesen. Unter Angabe des Klassennamens `Group` und des Gruppennamens löscht die `Delete()`-Methode die Gruppe aus dem Container.

ACHTUNG: Bitte verwechseln Sie nicht die Methoden `Remove()` und `Delete()`. `Remove()` entfernt einen Benutzer aus einer Gruppe. Eine Gruppe gilt nicht als ein Container-Objekt, weil die Gruppe den Benutzer im engeren Sinne nicht enthält, sondern nur einen Verweis auf den Benutzer speichert.

> Ein Objekt kann immer nur in einem Container sein. Wäre die Gruppe ein Container, könnte der Benutzer nur Mitglied einer einzigen Gruppe sein. Nach einem `Remove()` ist das Benutzerkonto immer noch vorhanden. `Delete()` dagegen entfernt einen Benutzer aus einem Container, sodass er permanent gelöscht wird.

Sofortige Löschung

Ein expliziter Aufruf von `SetInfo()` ist hier nicht notwendig. Die Löschung wird sofort durchgeführt.

Listing 9.9: /Skripte/Kapitel08/WinNT/LoescheGruppe.vbs

```
' LoescheGruppe.vbs
' Löschen einer Gruppe
' verwendet: ADSI
' ======================================================
Option Explicit
' Variablen deklarieren
Dim Container
' Konstanten definieren
Const GRUPPENNAME ="Finanzbeamte"
' Zugriff auf Domain-Objekt
Set Container = GetObject("WinNT://ServerE02")
' Löschen der Gruppe
Container.Delete "group", GRUPPENNAME
' Meldung ausgeben
Wscript.Echo "Gruppe " & GRUPPENNAME & " wurde gelöscht!"
```

9.1.9 Löschen eines Benutzers

Verwechslungen möglich

Ein Benutzer wird gelöscht durch den Aufruf der `Delete()`-Methode des Containers, in dem er enthalten ist. Das folgende Beispiel zeigt das Löschen eines Domänenbenutzers. Bei der `Delete()`-Methode ist – wie beim Erzeugen – der Klassenname `User` anzugeben, um Verwechslungen mit eventuell gleichnamigen `Group`-Objekten zu vermeiden. Der Aufruf von `SetInfo()` ist nicht notwendig; `Delete()` wird sofort ausgeführt!

Listing 9.10: /Skripte/Kapitel08/WinNT/LoescheBenutzer.vbs

```
' LoescheBenutzer.vbs
' Löschen eines Benutzerkontos
' verwendet: ADSI
' ==============================
Option Explicit
' Variable deklarieren
Dim Container
' Zugriff auf Domain-Objekt
Set Container = GetObject("WinNT://ServerE02")
' Benutzer löschen
Container.Delete "user", "WilliWinzig"
' Meldung ausgeben
WScript.Echo "Benutzer gelöscht!"
```

9.2 Active-Directory-Benutzerverwaltung unter Windows Server

Die folgenden Beispiele demonstrieren den Umgang mit der Benutzerverwaltung in einem Active Directory (kurz: AD) unter Windows Server (ab Version Windows 2000 Server bis einschließlich Windows Server 2016). Alle Beispiele setzen die im ersten Unterkapitel erzeugte Organisationseinheit *WSH-Scripting* voraus.

ACHTUNG: Weil dieser Fehler sehr häufig gemacht wird, seien Sie an dieser Stelle noch einmal gewarnt: Sie können mit den folgenden Skripten wirklich nur die Objekte in einem Active Directory im engeren Sinne verwalten. Benutzer und Gruppen, die lokal auf einem Windows-Client oder Windows-Server, der nicht Domänencontroller ist, existieren, gehören nicht zum Active Directory. Diese Objekte werden wie ein NT4-System verwaltet (vgl. vorheriges Unterkapitel). Gleiches gilt für einen Windows 2000 Server oder einen Windows Server 2003 (inkl. R2) oder einen Windows Server 2008 (inkl. R2), auf dem das Active Directory nicht installiert ist.

Microsoft hat in der Vergangenheit leider viele Benutzer mit einer zu globalen Verwendung des Begriffs Active Directory Service in die Irre geführt. Dies zeigt sich zum Beispiel auch am Namen der Komponente Active Directory Service Interface (ADSI). Wie bereits in Kapitel 5 geschildert, ist diese Komponente keineswegs nur für das Active Directory zuständig.

9.2.1 Anlegen einer Organisationseinheit

Verschachtelte Einheiten

Eine hervorstechende Eigenschaft des Active Directory (gegenüber der NT4-Benutzerverwaltung) besteht darin, beliebige Organisationsstrukturen in Form von Containern abbilden zu können. Ein solcher Container heißt im Active Directory `Organizational Unit`.

Oberste Ebene

Das Beispielskript erstellt eine Organisationseinheit im Active Directory. Dazu wird nach der Variablendeklaration ein Verweis auf das `RootDSE`-Objekt erzeugt und in der Variablen `Wurzel` abgelegt. `RootDSE` kennzeichnet das oberste Element in einem Active Directory.

ACHTUNG: Wichtig ist auch hier, dass Sie LDAP komplett in Großbuchstaben schreiben. Die Schreibweise der nachfolgenden Wörter ist jedoch egal (`rootdse`, `ROOTDSE`, `rootDSE` etc.).

Nun wird der LDAP-Pfad des Wurzelverzeichnisses vom Active Directory durch Abfrage der Eigenschaft `defaultNamingContext` ermittelt und in der Variablen `Domaene` gespeichert.

TIPP: Die Verwendung des RootDSE-Objekts bringt den Vorteil, dass Sie nicht den kompletten LDAP-Pfad zu Ihrem Active Directory wissen müssen. Selbstverständlich können Sie den Pfad aber auch manuell angeben. Dies ist in vielen der folgenden Skripte gezeigt.

Beachten Sie beim Anlegen von Organisationseinheiten den Klassennamen (`organizationalUnit`) im ersten Parameter und das dem eigentlichen Containernamen vorangestellte `OU=` im zweiten Parameter bei `Create()`.

Viele Eigenschaften

Die Organisationseinheiten stellen neben vielen Eigenschaften unter anderem die Eigenschaft `Description` zur Verfügung, die eine Beschreibung des Objekts zulässt. Erst durch den Aufruf von `SetInfo()` wird das Objekt im Active Directory abgelegt.

Listing 9.11: /Skripte/Kapitel08/LDAP/ErzeugeOU.vbs

```
' ErzeugeOU.vbs
' Erzeugen einer Organisationseinheit im Active Directory
' verwendet: ADSI
' ==============================
Option Explicit
' Variablen deklarieren
Dim Wurzel, Domaene, OrgEinheit
' Konstanten definieren
Const OUName="WSH-Scripting"
' Oberstes Element des AD holen
Set Wurzel = GetObject("LDAP://RootDSE")
' Wurzelverzeichnis bestimmen
Set Domaene = GetObject("LDAP://" & Wurzel.Get("defaultNamingContext"))
' OU anlegen
Set OrgEinheit = Domaene.Create("organizationalUnit", "OU=" & OUName)
' Beschreibung setzen
OrgEinheit.Description = "Dies ist eine OU für das Scripting-Buch"
' Werte festschreiben
OrgEinheit.SetInfo
' Ausgabe
WScript.Echo "OU wurde angelegt:" & OrgEinheit.AdsPath
```

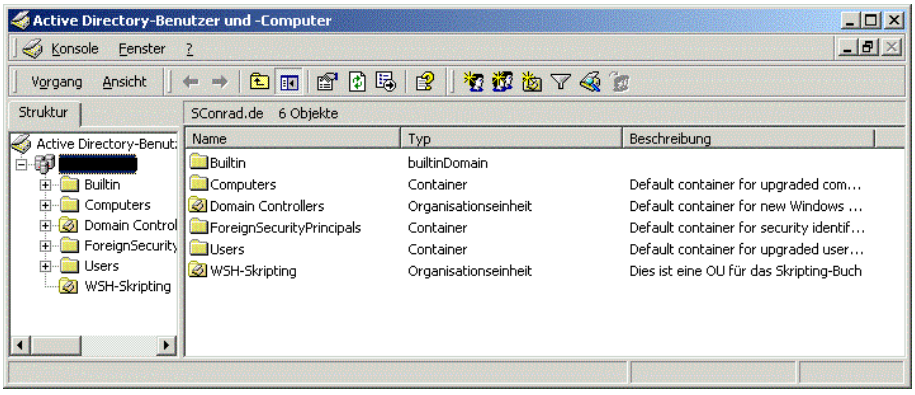

Abbildung 9.6: Die Organisationseinheit WSH-Scripting wurde erzeugt.

9.2.2 Anlegen eines Organisationseinheitenbaums im Active Directory

Untergeordnete Organisationseinheit anlegen

Durch den hierarchischen Aufbau ist das Active Directory in der Lage, `Container`-Objekte auf mehreren Ebenen aufzunehmen. Das Beispiel in diesem Abschnitt zeigt, wie man aus einer XML-Datei eine Organisationseinheitenhierarchie anlegt.

Das Finden der richtigen Organisationsstruktur ist oftmals ein Politikum beim Aufbau einer Active-Directory-Infrastruktur. Benötigt wird ein Instrument, mit dem man eine umfangreiche Hierarchie von Organisationseinheiten auf einfache Weise definieren und mit dessen Hilfe man die Organisationseinheiten nachher schnell implementieren kann.

Für hierarchische Daten hat sich der XML-Standard inzwischen durchgesetzt. Die folgende Abbildung zeigt, wie man eine Organisationsstruktur in XML-Form definieren könnte. Wenn man eine solche Hierarchie einmal definiert hat, liegt es nahe, ein Skript zu verwenden, das diese Hierarchie automatisch in das Active Directory einfließen lässt.

```
<?xml version="1.0" encoding="utf-8" ?>
- <OUStruktur>
  - <OU Name="www.IT-Visions.de">
      <OU Name="Geschäftsführung" />
    - <OU Name="Experten">
        <OU Name="DOTNET" />
        <OU Name="Scripting" />
      - <OU Name="Windows">
          <OU Name="Client" />
          <OU Name="Server" />
        </OU>
      </OU>
      <OU Name="Verwaltung" />
    </OU>
  </OUStruktur>
```

Abbildung 9.7: Eingabedatei für das Anlegen einer AD-OU-Struktur

9.2.2.1 Das Skript

Das Skript ist trotz seiner Mächtigkeit überschaubar. Neben dem Active Directory Service Interface (ADSI) kommt eine weitere Scripting-Komponente, das Microsoft XML Document Object Model (MSXML), zum Einsatz, um die XML-Datei zu lesen. Durch rekursive Programmierung (die Routine `ParseXMLDokument()` ruft sich immer wieder selbst auf, wenn es noch eine Unterebene gibt) kann man das Skript sehr prägnant halten. Das Ergebnis des Skripts ist in der nachfolgenden Bildschirmabbildung zu sehen.

Listing 9.12: /Skripte/Kapitel08/LDAP/OUs_AusXmlDateiAnlegen.vbs

```
' ----------------------------------------
' Skriptname: OUs_AusXmlDateiAnlegen.vbs
' Autor: Dr. Holger Schwichtenberg 2006-2007
' ----------------------------------------
' Dieses Skript erstellt eine OU-Struktur aus einer XML-Datei
' ----------------------------------------
' Verwendet SCRRun, MSXML, ADSI
' ----------------------------------------

Option Explicit
```

```
' Parameter
Const WURZEL = "LDAP://EO2/dc=it-visions,dc=local"
Const EINGABEDATEI = "OUStruktur.xml"

' Deklaration der Variablen
Dim Datei, WSHShell
Dim XMLDokument
Dim wurzelknoten

' COM-Objekte erstellen
Set WSHShell = CreateObject("Wscript.shell")
Set XMLDokument = CreateObject("Msxml2.DOMDocument")

' Pfad zur Eingabedatei
Datei = WSHShell.CurrentDirectory & "/" & EINGABEDATEI

' Asynchrones Laden ausschalten
XMLDokument.async = False
' Datei laden
XMLDokument.load(DATEI)
WScript.Echo "Dokument geladen"
' Fehler?
If XMLDokument.parseError.reason <> "" Then MsgBox XMLDokument.parseError.reason,
,"Fehler"
' Wurzel-Knoten auswählen
Set wurzelknoten = XMLDokument.selectSingleNode("OUStruktur")
' Durchlauf starten bei Wurzel
ParseXMLDokument 0, wurzelknoten, WURZEL

' === Alle Kind-Knoten durchlaufen
Sub ParseXMLDokument(ByVal ebene,  ByVal wurzelknoten, ByVal ouwurzel)
Dim OUKnoten, neuou
ebene = ebene + 1
For Each OUKnoten In wurzelknoten.Childnodes
WScript.echo "OU gefunden in XML-Datei: " & Space(ebene) & OUKnoten.
getAttribute("Name")
neuou = OUAnlegen(ouwurzel,OUKnoten.getAttribute("Name"))
ParseXMLDokument ebene, OUKnoten, neuou ' Rekursion
Next
End Sub

' === OU anlegen
Function OUAnlegen(Vater,OUName)
Dim Domain, OrgEinheit
' Verweis auf die bestehende OU holen
Set Domain = GetObject(Vater)
' Neue OU anlegen
Set OrgEinheit = Domain.Create("organizationalUnit", "OU=" & OUName)
' Beschreibung setzen
OrgEinheit.Description = "Angelegt mit dem Skript von Holger Schwichtenberg!"
' Werte festschreiben
OrgEinheit.SetInfo
' Ausgabe
WScript.Echo "OU wurde angelegt:" & OrgEinheit.ADsPath
OUAnlegen = OrgEinheit.ADsPath
End Function
```

Abbildung 9.8:
Ergebnis der Skriptausführung

9.2.3 Anlegen eines Benutzerkontos

Andere Attribute

Ähnlich, aber dennoch nicht identisch zu NT4 ist das Anlegen eines Benutzers im Active Directory. Neben dem Verzeichnisnamen benötigt jeder AD-Benutzer als Pflichtattribut einen `SAMAccountName`. Da bei `LDAP` anders als bei NT4 der Attributname des Schlüsselattributs (hier „`cn`“) Teil des Verzeichnisnamens ist, muss dem neuen Benutzernamen in der `Create()`-Methode getrennt durch ein Gleichheitszeichen der Attributname vorangestellt werden, der der Identifizierung der Instanzen dieser Klasse dient (also: „`cn=`“).

Benutzer in Container

Das Beispielskript legt einen neuen Benutzer innerhalb der Organisationseinheit *WSH-Scripting* an. Bitte haben Sie Verständnis dafür, dass nicht alle Attribute besprochen werden können, aber das würde den Rahmen dieses Buches sprengen. Die verwendeten Attribute sind:

- `SamAccountName`: kennzeichnet den Benutzer
- `AccountDisabled`: aktiviert oder deaktiviert das Konto des Benutzers

Listing 9.13: /Skripte/Kapitel08/LDAP/BenutzerAnlegen.vbs

```
' BenutzerAnlegen.vbs
' Erzeugen eines Benutzerkontos im AD
' verwendet: ADSI
' ==============================
Option Explicit
' Variablen deklarieren
Dim Container
Dim Benutzer
' Konstanten definieren
Const BenutzerName="HugoHastig"
' Bindung an Container
Set Container =
GetObject("LDAP://ServerEO2/OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Erzeugen des neuen Benutzers
Set Benutzer = Container.Create("user", "CN=" & BenutzerName)
' Attribute setzen
Benutzer.Put "samAccountName", BenutzerName
' Festschreiben der Werte
Benutzer.SetInfo
' Konto aktivieren
Benutzer.AccountDisabled = False
' Festschreiben der Werte
Benutzer.SetInfo
' Meldung ausgeben
WScript.Echo "Benutzer " & Benutzer.AdsPath & " angelegt"
```

Die folgende Darstellung zeigt den Benutzer innerhalb der Organisationseinheit „WSH-Scripting“.

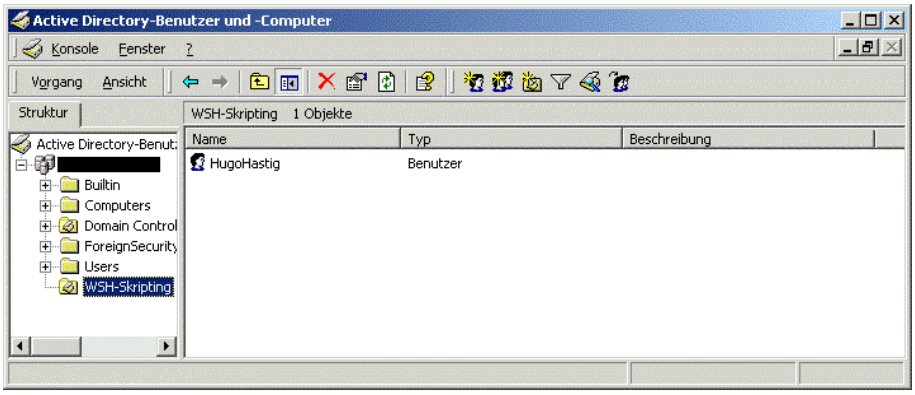

Abbildung 9.9: Der erzeugte Benutzer „HugoHastig“ in der Organisationseinheit „WSH-Scripting“

9.2.4 Anlegen von Benutzern aus einer Access-Datenbank

Große Netzwerke

Das Verwalten einzelner Benutzer kann noch in endlicher Zeit erledigt werden; kritisch wird die Verwaltung der Benutzer in großen Netzwerken. Als Datenbasis für das Anlegen von großen Benutzermengen eignet sich Microsoft Access bestens. Die Daten können über Formulare in einer Access-Anwendung verwaltet werden. Das folgende Skript liest alle in der Tabelle „Benutzer“ in der Datenbank *BenutzerDB.mdb* enthaltenen Benutzer aus und legt diese im Active Directory an.

Wiederverwendbarer Code

Zum Anlegen der Benutzer wird das Skript aus dem vorherigen Kapitel benutzt. Lediglich die Methode `SetPassword()` wird hier zusätzlich verwendet. Die Methode erlaubt das Setzen des Kennworts beim Anlegen eines Benutzerkontos (vgl. Kapitel 8.2.2).

HINWEIS: Die Verwendung von Access-Tabellen wird in Kapitel 6 beschrieben.

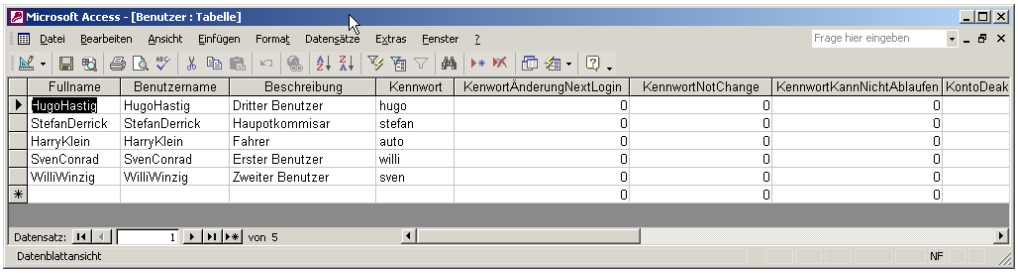

Abbildung 9.10: Die Tabelle „Benutzer“, die als Datenbasis dient

Listing 9.14: /Skripte/Kapitel08/LDAP/BenutzerAnlegenAusDatenbank.vbs

```
' BenutzerAnlegenAusDatenbank.vbs
' Erzeugen von AD-Benutzern aus einer Datenbank
' verwendet: ADSI, ADO
' ============================
Option Explicit
' Variablendeklaration
Dim DatenQuelle
Dim DBConnection, SqlString, Ergebnismenge
Dim Container, Benutzer
' Konstanten definieren
Const Verbindung="Provider=Microsoft.Jet.OLEDB.4.0; Data Source=BenutzerDB.MDB;"
' Connection-Objekt erzeugen
Set DBConnection = CreateObject("ADODB.Connection")
' Verbindung öffnen
DBConnection.Open Verbindung
' Alle Benutzer verwenden
SqlString="SELECT * FROM Benutzer"
' SQL-Statement ausführen
Set Ergebnismenge = DBConnection.Execute(SqlString)
On Error Resume Next
' An den Anfang des Abfrageergebnisses springen
Ergebnismenge.MoveFirst
' Bindung an Container
Set Container = GetObject("LDAP://ServerEO2/OU=WSH-Scripting,
                DC=IT-Visions,DC=de")
' Durchlaufe gesamte Datenbasis
Do While Not Ergebnismenge.eof
    ' Aufruf der Hilfsroutine
    BenutzerAnlegen Container, Ergebnismenge("Fullname"), _
                    Ergebnismenge("Kennwort")
    ' Nächsten Satz aus der Ergebnismenge holen
    Ergebnismenge.MoveNext
Loop
' Schließen der Abfrage
Ergebnismenge.Close
' Schließen der Verbindung
DBConnection.Close

Sub BenutzerAnlegen (Container, Benutzername,Passwort)
' Hilfsroutine: Erzeugen eines Benutzerkontos unter Windows 2000
' Variablen deklarieren

Dim Benutzer
' Erzeugung des neuen Benutzers
Set Benutzer = Container.Create("user", "cn=" & Benutzername)
' Attribute setzen
Benutzer.Put "samAccountName", CStr(Benutzername)
' Festschreiben der Werte
Benutzer.SetInfo
' Konto aktivieren
Benutzer.AccountDisabled = false
Benutzer.SetInfo
' Kennwort des Benutzers setzen
Benutzer.SetPassword Passwort
' Meldung ausgeben
WScript.Echo "Benutzer " & Benutzer.AdsPath & " angelegt"
```

```
 ' Freigeben der Objekte
End Sub
```

9.2.5 Anlegen einer Benutzergruppe

Um Benutzer in Gruppen verwalten zu können, müssen diese erst angelegt werden. Eine Zuweisung von Benutzern an nicht vorhandene Gruppen erzeugt den Laufzeitfehler „Ein solches Objekt ist auf dem Server nicht vorhanden.“.

Kompatibilität

Das nachfolgende Skript generiert nach der Konstantendefinition und Variablendeklaration ein `Domain`-Objekt durch den Zugriff auf das LDAP-Verzeichnis. Die `Create()`-Methode erzeugt ein `Group`-Objekt und legt es in der Variablen `Gruppe` ab. Dem Namen der Gruppe muss `cn=` vorangestellt werden.

Im nächsten Schritt werden zwei Eigenschaften des `Group`-Objekts gesetzt. Die Eigenschaft `samAccountName` kennzeichnet den Gruppennamen sowohl für die Windows-NT-3.51- bzw. -4.0-Welt als auch für das Active Directory.

Group Type

Die Eigenschaft `GroupType` gibt den Gruppentyp an. Für die Gruppentypen existieren nachfolgende Werte, die bitweise verknüpft werden können (`OR`-Operator):

```
Const GLOBAL_GROUP = 2
Const LOCAL_GROUP = 4
Const UNIVERSAL_GROUP = 8
Const SECURITY_ENABLED = -2147483648
```

Beispiele:

- 4 ist eine lokale Verteilergruppe.
- `-2147483644` (`=4 or -2147483648`) ist eine lokale Sicherheitsgruppe.

Der Aufruf von `SetInfo()` schreibt die Änderungen im Active Directory fest.

Listing 9.15: /Skripte/Kapitel08/LDAP/GruppeAnlegen.vbs

```
' GruppeAnlegen.vbs
' Erzeugen einer Gruppe im Active Directory
' verwendet: ADSI
' ================================================================
Option Explicit
' Konstantendefinition
Const ADS_GROUP_TYPE_DOMAIN_LOCAL_GROUP = &H4
Const ADS_GROUP_TYPE_SECURITY_ENABLED = &H80000000
Const GruppenName="ScriptingGruppe"
' Variablendeklaration
Dim Gruppe
Dim Domaene
' Domain-Objekt erzeugen
Set Domaene = GetObject _
("LDAP://ServerEO2/OU=WSH-Scripting,DC=IT-Visions,DC=de")
'Erzeugen des Group-Objekts
Set Gruppe = Domaene.Create("group", "CN=" & GruppenName)
' Name für WinNT
Gruppe.Put "samAccountName", GruppenName
```

```
' Gruppentyp setzen
Gruppe.Put "groupType", ADS_GROUP_TYPE_DOMAIN_LOCAL_GROUP _
Or ADS_GROUP_TYPE_SECURITY_ENABLED
' Werte festschreiben
Gruppe.SetInfo
' Ausgabe
WScript.echo "Die Gruppe " & Gruppe.AdsPath & " wurde angelegt."
```

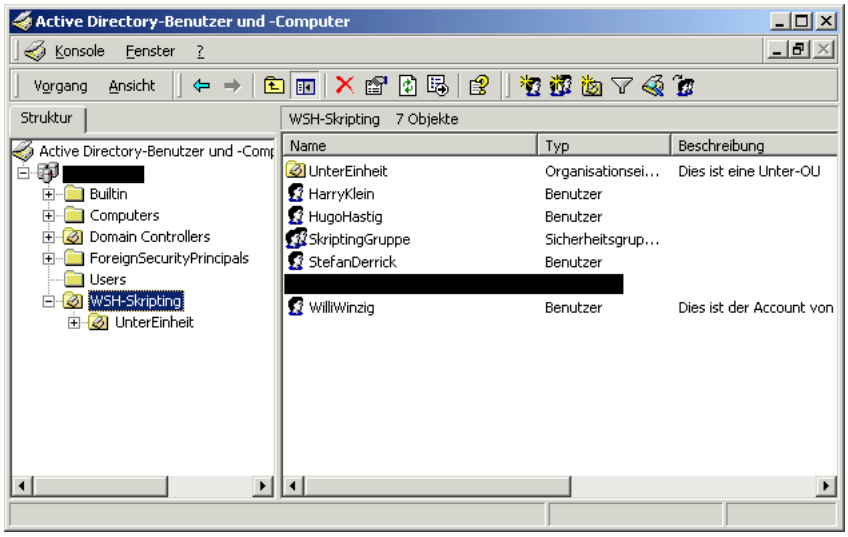

Abbildung 9.11: Die Gruppe „ScriptingGruppe“ wurde angelegt.

TIPP: Das Attribut `samAccountName` und der Name der Gruppe müssen nicht gleich sein. Hier sind unterschiedliche Zuweisungen möglich.

9.2.6 Hinzufügen eines Benutzers einer Gruppe

Rechte-zuweisung

Das Verwalten von Benutzern in Gruppen erleichtert dem Administrator die Zuweisung von Rechten an eine Auswahl von Personen. Ein Benutzer kann einer beliebigen Anzahl von Gruppen zugeordnet werden. Das Beispielskript demonstriert eine solche Zuordnung eines Benutzers zu einer Gruppe.

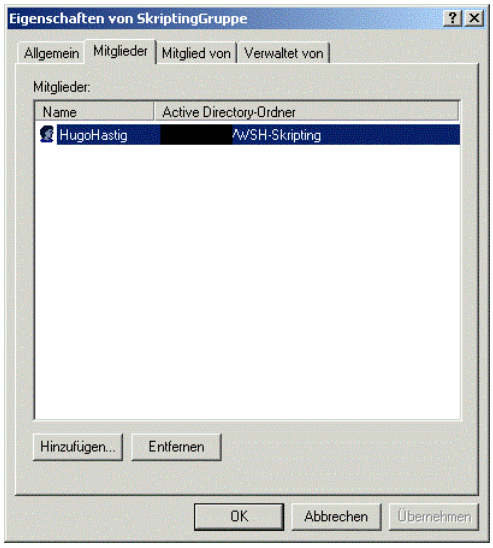

Abbildung 9.12:
Benutzer „HugoHastig“ zur „ScriptingGruppe“ hinzugefügt

LDAP-Identifikation

Durch `GetObject()` wird eine Referenz auf das `Group`-Objekt der angegebenen Gruppe erzeugt und anschließend wird mittels `Add()` der Benutzer der Gruppe zugeordnet. Hierbei ist sowohl die Identifikation der Gruppe als auch die des Benutzers über den kompletten LDAP-Pfad notwendig.

Listing 9.16: /Skripte/Kapitel08/LDAP/BenutzerzuGruppe.vbs

```
' BenutzerzuGruppe.vbs
' Hinzufügen eines Benutzerkontos zu einer Gruppe im AD
' verwendet: ADSI
' ===================================================================
Option Explicit
' Variablen deklarieren
Dim Gruppe
' Konstanten definieren
Const GruppenName="ScriptingGruppe"
Const BenutzerName="HugoHastig"
' Bindung an Gruppen-Container
Set Gruppe = GetObject("LDAP://ServerEO2/CN=" & GruppenName & _
",OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Hinzufügen eines Benutzers zur Gruppe
Gruppe.Add ("LDAP://ServerEO2/CN=" & BenutzerName & _
",OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Ausgabe
WScript.Echo "Der Benutzer " & BenutzerName & " wurde der Gruppe " _
 & GruppenName & " hinzugefügt."
```

9.2.7 Ändern des Kennworts

Das Ändern von Kennwörtern ist eine der am häufigsten vergessenen Aufgaben eines Benutzers. In diesem Beispiel wird dem Administrator ein Skript an die Hand gegeben, mit dem Kennwörter von Benutzern geändert werden können.

Wie NT 4.0

Auch hier werden wie bei der Benutzerverwaltung von Windows NT 4.0 zwei Methoden für den Kennwortwechsel angeboten:

- Mit der Methode `ChangePassword()` kann das Kennwort nur unter Angabe des bisherigen Kennworts geändert werden.
- Bei `SetPassword()` ist die Angabe des bisherigen Kennworts nicht erforderlich.

TIPP: Bitte beachten Sie auch die Hinweise zur Kennwortänderung bei der NT4-Benutzerverwaltung (Kapitel 8.2.2).

Sofortige Änderung

Im Skript wird durch `GetObject()` ein Verweis auf das Benutzerkonto erzeugt, dessen Kennwort geändert werden soll. Als Parameter ist der komplette LDAP-Pfad auf das `User`-Objekt anzugeben. Anschließend erfolgt der Aufruf der Methode `SetPassword()`, die das Kennwort auf den neuen Wert setzt. Diese Änderung wird sofort durchgeführt.

Listing 9.17: /Skripte/Kapitel08/LDAP/KennwortAendern.vbs

```
' KennwortAendern.vbs
' Ändern eines Benutzerkennwortes im AD
' verwendet: ADSI
' ==============================
Option Explicit
' Variablen deklarieren
Dim Container
Dim Benutzer
' Konstanten definieren
Const BenutzerName="HugoHastig"
Const Kennwort="williw"
' Zugriff auf das Benutzer-Objekt
Set Benutzer = GetObject("LDAP://laptop/CN=" & BenutzerName & _
",OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Kennwort ändern
Benutzer.SetPassword(Kennwort)
' Meldung ausgeben
WScript.Echo "Kennwort für Benutzer " & Benutzer.AdsPath & " wurde geändert"
```

9.2.8 Umbenennen eines Benutzers

Nicht über Attribute

Die Umbenennung eines Benutzers wird in diesem Beispiel anhand des folgenden Skripts demonstriert. Dazu wird die Methode `MoveHere()` verwendet.

HINWEIS: Eine Umbenennung über die Zuweisung an die im User-Objekt vorhandenen Attribute ist nicht möglich. So ist es nicht möglich, den Namen, der in der „Active Directory-Benutzer und -Computer"-Konsole angezeigt wird, zu verändern, sondern lediglich die Eigenschaften `givenName`, `samAccountName` und `displayName`. Keines dieser drei Attribute steuert allerdings die Anzeige in besagter Konsole.

Umbenennen durch Verschieben

Um nun einen Benutzer umzubenennen, wird erst ein Verweis auf ein bestehendes Benutzerkonto erzeugt. Dies geschieht durch die Zuweisung des kompletten LDAP-Pfads und den anschließenden Aufruf von `GetObject()`. Nun werden der `MoveHere()`-Methode der LDAP-Pfad zu dem zu ändernden Benutzerkonto sowie der neue Name des Benutzers übergeben.

Listing 9.18: /Skripte/Kapitel08/LDAP/BenutzerUmbenennen.vbs

```
' BenutzerUmbenennen.vbs
' Umbenennen eines Benutzers im Active Directory
' verwendet: ADSI
' ==============================
Option Explicit
' Variablen deklarieren
Dim Domaene
' Konstanten definieren
Const NeuerName="WilliRiesig"
Const BenutzerName="HugoHastig"
' Bindung an Domain-Container
Set Domaene = GetObject("LDAP://ServerEO2/OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Setzen der Attribute
Domaene.MoveHere "LDAP://ServerEO2/CN=" & BenutzerName & _
                 ",OU=WSH-Scripting,DC=IT-Visions,DC=de","CN=" & NeuerName
Wscript.Echo "Benutzer wurde umbenannt"
```

ACHTUNG: Eine Gesamtübersicht über die Attribute des User-Objekts würde den Rahmen dieses Kapitels sprengen. Um einen Überblick über die einzelnen Attribute von Objekten zu erhalten, hat sich der Microsoft Active Directory Service Browser (ADB) als sehr nützlich erwiesen. Sie finden ihn in den Downloads zu diesem Buch im Verzeichnis */Install/Werkzeuge/ADSI*.

9.2.9 Ändern der Benutzerdaten

Datenänderung

Das folgende Skript zeigt, wie man verschiedene Eigenschaften eines vorhandenen Benutzers ändern kann. Die Attributnamen sind sprechend und daher verständlich.

Listing 9.19: /Skripte/Kapitel08/LDAP/BenutzerdatenAendern.vbs

```
' BenutzerdatenAendern.vbs
' Ändern von Benutzerdaten im Active Directory
' verwendet: ADSI
```

```
' ==============================
Option Explicit
' Variablen deklarieren
Dim Benutzer
' Bindung an Benutzer-Container
Set Benutzer = GetObject("LDAP://ServerEO2/CN=WilliRiesig, _
OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Setzen der Attribute
Benutzer.Put "samAccountName", "WilliRiesig"
Benutzer.Put "givenName", "Willi"
Benutzer.Put "sn", "Riesig"
Benutzer.Put "displayName", "WilliRiesig"
Benutzer.Put "physicalDeliveryOfficeName", "Zimmer 4711"
Benutzer.Put "telephoneNumber", "555-789877"
Benutzer.Put "mail", "williRiesig@IT-Visions.de"
Benutzer.Put "description", "Dies ist der Account von Willi Riesig"
Benutzer.Put "WWWHomePage", "http://www.IT-Visions.de"
' Werte werden festgeschrieben
Benutzer.SetInfo
Wscript.Echo "Benutzer " & Benutzer.AdsPath & " wurde geändert!"
```

9.2.10 Deaktivieren eines Benutzerkontos

Sperrung

Soll einem Benutzer der Zugang zum Netzwerk nur kurzfristig entzogen werden, muss man das Konto nicht löschen, sondern kann es kurzfristig deaktivieren.

Das nachfolgende Beispiel zeigt, wie mithilfe des Attributs `userAccountControl` ein Benutzer deaktiviert wird, sodass er sich nicht mehr am Netz anmelden kann. Dazu wird mit `GetObject()` ein Verweis auf das `User`-Objekt erstellt und in der Variablen `Benutzer` gespeichert. Die Referenz auf den Benutzer erfordert den kompletten LDAP-Pfad zur Identifikation.

Deaktivierung

Nun wird in der Variablen `UserAccountData` der aktuelle Wert der Eigenschaft `userAccountControl` abgelegt. Durch Verknüpfung des aktuellen Status von `userAccountControl` mit dem Wert der Konstanten `ADS_UF_ACCOUNTDISABLE` (2) über den `OR`-Operator und die Zuweisung des Werts mittels `Put()` wird das Konto zur Sperrung vorbereitet. Die eigentliche Sperrung erfolgt erst mit dem Aufruf von `SetInfo()`.

Listing 9.20: /Skripte/Kapitel08/LDAP/DeaktiviereKonto.vbs

```
' DeaktiviereKonto.vbs
' Deaktivieren eines Benutzerkontos im Active Directory
' verwendet: ADSI
' ==============================
Option Explicit
' Variablen deklarieren
Dim Container
Dim Benutzer
Dim UserAccountData
' Konstanten definieren
Const BenutzerName="WilliRiesig"
Const ADS_UF_ACCOUNTDISABLE = 2
' Zugriff auf das Benutzer-Objekt
```

```
Set Benutzer = GetObject("LDAP://ServerEO2/CN=" & BenutzerName & _
               ",OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Benutzerdaten ermitteln
UserAccountData = Benutzer.Get("userAccountControl")
' Daten ändern
Benutzer.Put "userAccountControl", UserAccountData OR ADS_UF_ACCOUNTDISABLE
' Änderungen festschreiben
Benutzer.SetInfo
' Meldung ausgeben
WScript.Echo "Benutzerkonto " & Benutzer.AdsPath & " wurde deaktiviert"
```

HINWEIS: Das Aktivieren des gesperrten Kontos ist ebenfalls möglich. Allerdings existiert hierfür keine Konstante. Das Konto kann durch die Verknüpfung der Eigenschaft `userAccountControl` mit dem Wert 4 durch den AND-Operator wieder aktiviert werden. Auch hier ist der Aufruf von `SetInfo()` notwendig.

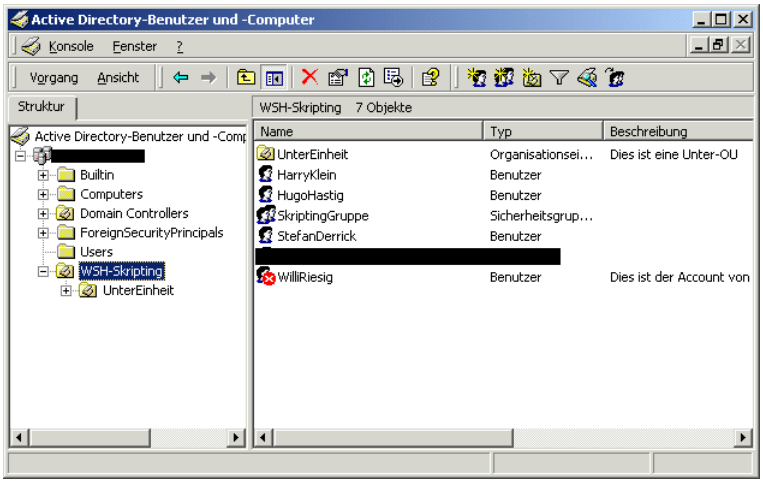

Abbildung 9.13: Benutzerkonto für „WilliRiesig“ ist deaktiviert.

9.2.11 Entfernen eines Benutzers aus einer Gruppe

Rechte-entzug

Durch Entfernen eines Benutzers aus einer Gruppe können Rechte und Beschränkungen bequem entfernt bzw. hinzugefügt werden. Im Beispiel wird ein Verweis auf das `Group`-Objekt der angegebenen Gruppe erzeugt und durch die `Remove()`-Methode des `Group`-Objekts wird der Benutzer aus der Gruppe entfernt. Sowohl die Identifikation des `Group`-Objekts als auch die des `User`-Objekts erfolgt über einen gültigen LDAP-Pfad.

Listing 9.21: /Skripte/Kapitel08/LDAP/LoescheBenutzerausGruppe.vbs

```
' LoescheBenutzerausGruppe.vbs
' Löschen eines Benutzerkontos aus einer Gruppe im AD
```

```
' verwendet: ADSI
' ==============================
Option Explicit
' Variablen deklarieren
Dim Gruppe
' Konstanten definieren
Const GruppenName="ScriptingGruppe"
Const BenutzerName="HugoHastig"
' Bindung an Gruppen-Container
Set Gruppe = GetObject("LDAP://ServerEO2/CN=" & GruppenName & _
",OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Entfernen des Benutzers
Gruppe.Remove ("LDAP://ServerEO2/CN=" & BenutzerName & _
",OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Ausgabe
WScript.Echo "Der Benutzer " & BenutzerName & _
 " wurde aus der Gruppe " & GruppenName & " entfernt."
Löschen einer Gruppe
```

Werden Gruppen nicht mehr benötigt, weil beispielsweise Abteilungen aufgelöst wurden, lassen sich diese Objekte auch wieder aus dem Active Directory entfernen.

Gruppenlöschung

Das Beispiel demonstriert das Vorgehen an der aus den vorherigen Kapiteln bekannten Gruppe „ScriptingGruppe“. Um diese Gruppe zu löschen, wird durch `GetObject()` ein Verweis auf das `Users`-Objekt angelegt. Ein anschließender Aufruf der `Delete()`-Methode löscht die Gruppe. Als Parameter werden der Klassenname `Group` sowie der Name des `Group`-Objekts (mit vorangestelltem `CN=`) erwartet.

HINWEIS: Die `Delete()`-Methode gibt kein Objekt zurück, und ein Aufruf von `SetInfo()` ist nicht notwendig, da die Aktion sofort ausgeführt wird.

Listing 9.22: /Skripte/Kapitel08/LDAP/LoescheGruppe.vbs

```
' LoescheGruppe.vbs
' Entfernen einer Gruppe im AD
' verwendet: ADSI
' ================================================================
Option Explicit
' Variablendeklaration
Dim Domaene
' Konstanten definieren
Const GruppenName="ScriptingGruppe"
' Domain-Objekt erzeugen
Set Domaene = GetObject _
("LDAP://ServerEO2/OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Löschen der Gruppe
Domaene.Delete "group", "CN=" & GruppenName
' Ausgabe
WScript.echo "Die Gruppe " & GruppenName & " wurde entfernt."
```

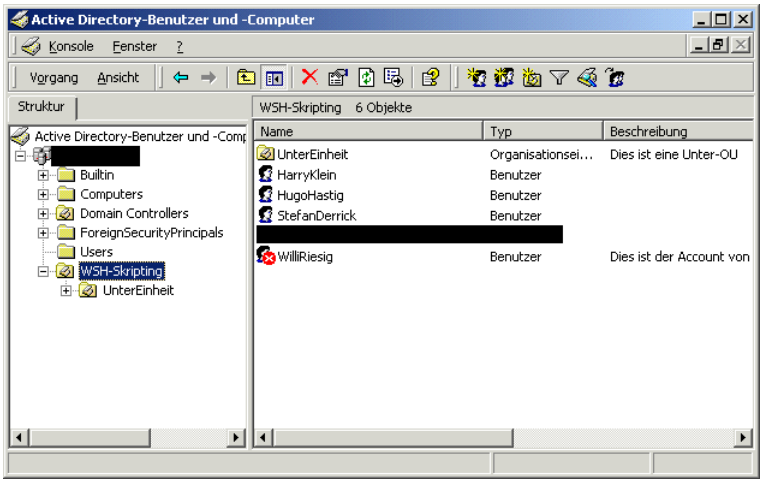

Abbildung 9.14: Die Gruppe wurde gelöscht.

9.2.12 Löschen eines Benutzerkontos

Benutzer loswerden

Da nicht alle Benutzer im Unternehmen verbleiben, muss gelegentlich auch mal einer gelöscht werden. Um nun einen angelegten Benutzer wieder loszuwerden, nutzt das Beispielskript die `Delete()`-Methode des übergeordneten Containers. Der Verweis auf den Container, in diesem Fall die Organisationseinheit *WSH-Scripting*, wird über den LDAP-Pfad referenziert und durch den Aufruf von `GetObject()` erzeugt. Anschließend wird die `Delete()`-Methode mit dem Klassennamen `User` und dem Namen des Benutzers (mit vorangestelltem `CN=`) aufgerufen. Die `Delete()`-Methode löscht den Benutzer sofort und ohne Nachfrage.

Listing 9.23: /Skripte/Kapitel08/LDAP/LoescheBenutzer.vbs

```
' LoescheBenutzer.vbs
' Löschen eines Benutzerkontos im AD
' verwendet: ADSI
' ==============================
Option Explicit
' Variablen deklarieren
Dim Container
' Konstanten definieren
Const BenutzerName="HugoHastig"
' Zugriff auf das Container-Objekt
Set Container = GetObject _
("LDAP://ServerEO2/OU=WSH-Scripting,DC=IT-Visions,DC=de")
' Benutzer löschen
Container.Delete "User", "CN=" & BenutzerName
' Meldung ausgeben
WScript.Echo "Benutzer " & Benutzername & " wurde gelöscht"
```

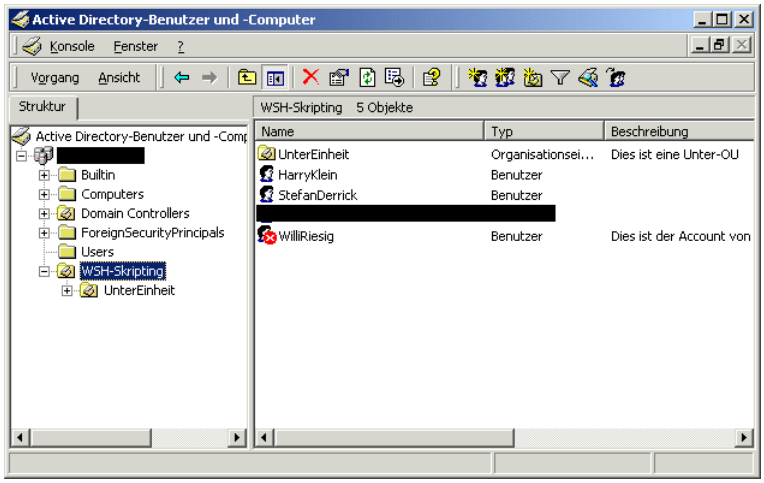

Abbildung 9.15: Der Benutzer HugoHastig wurde gelöscht.

9.2.13 Löschen einer Organisationseinheit

Container löschen

Analog zur Erstellung von Organisationseinheiten lassen sich diese auch wieder aus dem Active Directory entfernen. Die Vorgehensweise ist identisch mit dem Löschen eines Benutzers oder einer Gruppe. Anstelle der `Create()`-Methode wird zum Löschen die `Delete()`-Methode aufgerufen.

ACHTUNG: Die Organisationseinheit kann nur gelöscht werden, wenn sie keine Unterobjekte mehr enthält.

Listing 9.24: /Skripte/Kapitel08/LDAP/LoescheOU.vbs

```
' LoescheOU.vbs
' Löschen einer Organisationseinheit im AD
' verwendet: ADSI
' ================================
' Variablendeklaration
Dim Root
Dim Domain
' Konstanten definieren
Const OUName="WSH-Scripting"
' Oberstes Element des AD holen
Set Wurzel = GetObject("LDAP://RootDSE")
' Wurzelverzeichnis bestimmen
Set Domaene = GetObject("LDAP://" & Wurzel.Get("defaultNamingContext"))
' OU löschen
Domaene.Delete "organizationalUnit", "OU=" & OUName
' Ausgabe
WScript.Echo "OU " & OUName & " wurde gelöscht"
```

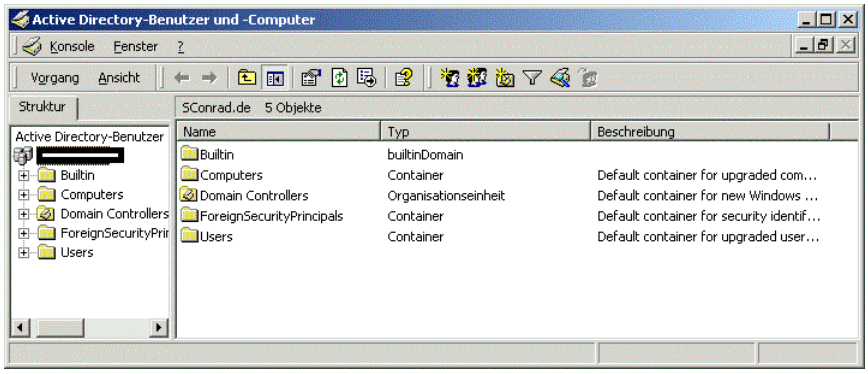

Abbildung 9.16: Nun ist die OU auch wieder weg.

9.3 Fragen und Aufgaben

1. Verliert ein Benutzer nach der Umbenennung seines Kontos seine zugewiesenen Berechtigungen?
2. Wird beim Zuordnen eines Benutzers zu einer Gruppe dieser Benutzer automatisch angelegt, wenn er nicht vorhanden ist?
3. Aktiviert eine Zuweisung des Werts `True` an die Eigenschaft `AccountDisabled` das Konto eines Benutzers oder deaktiviert sie es?
4. Können Benutzer in der NT4-Benutzerdatenbank wie im Active Directory auf andere Ebenen verschoben werden?
5. Wenn eine Gruppe gelöscht wird, werden dann automatisch die darin enthaltenen Benutzer gelöscht?
6. Ist eine Organisationseinheit ein Container oder ein Blatt-Objekt?
7. Kann das Kennwort eines Benutzers ohne Kenntnis des alten Kennworts geändert werden? Wenn ja, mit welcher Methode?
8. Was kennzeichnet der nachfolgende LDAP-Pfad genau? *LDAP://ServerE02/CN=Trainer, OU=Scripting,DC=IT-Visions,DC=de*
9. Ist die Verschachtelung von Container-Objekten einer Beschränkung unterworfen?
10. Lassen sich Container-Objekte mit einem einfachen `Delete()` rekursiv löschen?

Stichwortverzeichnis

H

I

N

O

P

T

U

V

W

X

Y

Z