K A P I T E L 2

Verwalten und Automatisieren von SharePoint

In Kapitel 1, »Erstellen eines SharePoint 2010-Intranets«, haben Sie die Zentraladministration verwendet, um alltägliche Verwaltungsarbeiten durchzuführen, die mit der Installation und Konfiguration von Microsoft SharePoint Server 2010 zu tun haben.

In diesem Kapitel erfahren Sie mehr drüber, was es bedeutet, Administrator einer SharePoint-Farm zu sein und SharePoint mit der Zentraladministration und auf der Befehlszeile zu verwalten.

Zu den leistungsfähigsten Werkzeugen, die Ihnen als SharePoint-Administrator zur Verfügung stehen, gehört Windows PowerShell. SharePoint 2010 bietet eine umfangreiche Unterstützung für Windows PowerShell als primäres Befehlszeilenwerkzeug für die Verwaltung und Automatisierung von SharePoint. In diesem Kapitel lernen Sie die Grundlagen von Windows PowerShell für SharePoint kennen.

In diesem Kapitel abgedeckte Prüfungsziele:

- Verwalten der Konten und Benutzerrollen
- Bereitstellen von Neuinstallationen und Aktualisierungen
- Konfigurieren von SharePoint-Farmen
- Verwalten von Webanwendungen
- Verwalten von Websitesammlungen

Lektionen in diesem Kapitel:

- Lektion 1: SharePoint-Verwaltungsrollen und Werkzeuge 81
- Lektion 2: Automatisieren von SharePoint-Vorgängen mit Windows PowerShell . . 94

Bevor Sie beginnen

Um die Lektionen in diesem Kapitel durchführen zu können, müssen Sie die Übungen in Kapitel 1 durchgeführt und abgeschlossen haben.

Praxistipp

Dan Holme

Dieses Kapitel behandelt Themen, die für viele SharePoint-Administratoren zu den verwirrendsten und unverständlichsten Konzepten und Fertigkeiten zählen. SharePoint 2010 (aus technischer Sicht handelt es sich um SharePoint Version 4) ist bei einem Vergleich mit älteren Versionen kaum wiederzuerkennen. Das Erbe dieser Vorgängerversionen lässt sich allerdings nicht so einfach abschütteln, denn die Terminologie der zugrunde liegenden Maschine – das Objektmodell und die Befehlszeilenparameter – unterscheiden sich sehr von der Terminologie, die in der Benutzeroberfläche und der Dokumentation verwendet wird. Lektion 1 beschreibt die Verwaltungsrollen und die Werkzeuge, die Ihnen zur Verwaltung von SharePoint zur Verfügung stehen, nämlich die Zentraladministration, *Stsadm.exe* und Windows PowerShell. Lektion 2 behandelt die wahrscheinlich wichtigste Änderung in SharePoint 2010, nämlich die Einführung von Windows PowerShell als bevorzugtes Werkzeug für die Arbeit auf der Befehlszeile und die Automatisierung. SharePoint hat lange unter einem Mangel an Werkzeugen gelitten, mit denen man über mehrere Websites, Websitesammlungen und Webanwendungen hinweg arbeiten und die Systeme überprüfen und überwachen kann. Windows PowerShell füllt diese Lücke und ermöglicht es Administratoren, produktiver zu sein als nur mit der Zentraladministration und *Stsadm.exe*.

Lektion 1: SharePoint-Verwaltungsrollen und Werkzeuge

Bisher haben Sie SharePoint nur mit einem sehr bevorrechtigten Konto installiert und konfiguriert, nämlich mit *SP_Admin*, das Mitglied der SharePoint-Gruppe *Farm Administrators* und der lokalen Gruppe *Administratoren* auf einem SharePoint-Server ist. Und Sie haben die meisten Konfigurationen mit der Zentraladministration durchgeführt. In dieser Lektion verfeinern Sie das Verwaltungsmodell und delegieren Verwaltungsrollen an bestimmte Benutzer. Sie lernen die Arbeit mit *Stsadm.exe* kennen, eines der beiden wichtigsten Befehlszeilenprogramme von SharePoint, und Sie erkunden die Zentraladministration selbst.

Am Ende dieser Lektion werden Sie in der Lage sein, die folgenden Aufgaben auszuführen:

- Beschreiben und Verwalten der SharePoint-Verwaltungsrollen
- Beschreiben der Optionen zur Verwaltung von SharePoint-Farmen
- Verwalten von SharePoint auf der Befehlszeile mit *Stsadm.exe*
- Konfigurieren und Verwalten der Zentraladministration

Veranschlagte Zeit für diese Lektion: 45 Minuten

Verwaltungsrollen

In älteren SharePoint-Versionen war es schwierig, die Verwaltung von bestimmten Diensten oder Einstellungen an andere Benutzer zu delegieren. SharePoint 2010 bietet feiner abgestufte Verwaltungsrollen:

- Farmadministratoren
- Windows-Administratoren
- Dienstanwendungsadministratoren
- Dienstanwendungsfeatureadministratoren
- Websitesammlungsbesitzer
- Websitesammlungsadministratoren
- Websitegruppen

Diese Rollen werden in den folgenden Abschnitten näher beschrieben.

Farmadministratoren

Wie Sie aus Kapitel 1 wissen, versammelt die Gruppe *Farm Administrators* die Konten, die mit der Zentraladministration Verwaltungsarbeiten durchführen können. Standardmäßig sind das Setup-Benutzerkonto (beispielsweise *SP_Admin*), das Farmkonto (*SP_Farm*) und die lokale Gruppe *Administratoren* Mitglieder der Gruppe *Farm Administrators*. In Kapitel 1 wurden Sie auch darauf hingewiesen, dass Sie nach der Installation von SharePoint die geeigneten Benutzerkonten zur Gruppe *Farm Administrators* hinzufügen sollten, beispielsweise Ihr eigenes Benutzerkonto. Anschließend sollten Sie sich mit diesem Verwaltungskonto anmelden und nicht mehr als *SP_Admin*. Natürlich können Sie *SP_Admin* noch verwenden, um größere Änderungen auf Produkt- oder Farmebene durchzuführen und beispielsweise Produkte, Sprachpakete oder Service Packs zu installieren oder Server zur Farm hinzuzufügen oder aus ihr zu entfernen.

Verwalten der Mitgliedschaft in der Farmadministratorgruppe

1. Klicken Sie in der Schnellstartleiste der Zentraladministration auf *Sicherheit*.
2. Klicken Sie im Abschnitt *Benutzer* auf *Farmadministratorgruppe verwalten*.

Mitglieder der Farmadministratorgruppe haben die Verantwortung für alle Server aus der Serverfarm und verfügen über entsprechende Berechtigungen. Sie können in der Zentraladministration alle Verwaltungsarbeiten für die Server oder die Serverfarm durchführen. Mitglieder dieser Gruppe können auch Windows PowerShell verwenden, um Konfigurationsdatenbankobjekte zu erstellen und zu verwalten, und sie können Befehlszeilenprogramme wie *Stsadm.exe* verwenden. Außerdem können sie Administratoren für die Verwaltung von Dienstanwendungen (Instanzen von gemeinsamen Diensten) einsetzen.

Die Farmadministratorgruppe verfügt standardmäßig nicht über die Berechtigung für den Zugriff auf einzelne Websites oder deren Inhalte. Allerdings können ihre Mitglieder eine Websitesammlung in Besitz nehmen, indem sie sich selbst in der Zentraladministration als Websitesammlungsadministrator einsetzen. Verlässt zum Beispiel ein Websitesammlungsadministrator die Firma und es muss ein neuer Administrator eingesetzt werden, können Mitglieder der Gruppe *Farm Administrators* die Websitesammlung in Besitz nehmen und die Änderung durchführen.

Windows-Administratoren

Mitglieder der Gruppe *Administratoren* eines SharePoint-Servers, auf dem die Zentraladministration ausgeführt wird, sind automatisch auch Mitglieder der Gruppe *Farm Administrators*. Daher können Mitglieder der Gruppe *Administratoren* des lokalen Servers alle Farmadministratoraufgaben durchführen. Außerdem können lokale Administratoren neue Produkte oder Anwendungen installieren, Webparts und neue Features im Global Assembly Cache bereitstellen, neue Webanwendungen und neue IIS-Websites (Internetinformationsdienste) erstellen und Dienste starten.

Wie Farmadministratoren haben auch Mitglieder der Gruppe *Administratoren* des lokalen Servers standardmäßig keinen Zugriff auf Websiteinhalte, können aber eine Websitesammlung in Besitz nehmen und sich auf diese Weise selbst Zugriff auf Inhalte geben.

Empfohlene Vorgehensweise **Verwalten der Gruppe *Administratoren***

Sorgen Sie dafür, dass möglichst wenige Benutzer auf einem Server Mitglieder der Gruppe *Administratoren* sind, um das Risiko der versehentlichen oder absichtlichen Beschädigung von Inhalten und eines Ausspähens von Daten zu minimieren. Außerdem ist es wichtig, dass die Mitgliedschaft in der Gruppe *Administratoren* auf allen SharePoint-Servern einer Farm einheitlich gehandhabt wird.

Dienstanwendungsadministratoren

In Kapitel 1 haben Sie Dienstanwendungen als logische Verwaltungsinstanzen von SharePoint-Diensten wie zum Beispiel dem Suchdienst kennengelernt. Viele Dienstanwendungen haben Konfigurationen, die verwaltet werden müssen. Farmadministratoren haben die Berechtigung, alle Dienstanwendungen zu verwalten. Diese Berechtigungen können ihnen nicht genommen werden. In SharePoint 2007 war es nicht einfach, die Verwaltung und Konfiguration eines einzelnen Dienstes zu delegieren, aber in SharePoint 2010 können Sie die Verwaltung von Dienstanwendungen delegieren.

Dienstanwendungsadministratoren werden durch Mitglieder der Gruppe *Farm Administrators* eingesetzt. Die Administratoren einer Dienstanwendung können Einstellungen für eine bestimmte Dienstanwendung in der Farm vornehmen. Allerdings können sie keine Dienstanwendungen erstellen, nicht auf andere Dienstanwendungen aus der Farm zugreifen und keine Vorgänge auf Farmebene durchführen, auch keine Topologieänderungen. Der Dienstanwendungsadministrator für eine Suchdienstanwendung aus einer Farm kann zum Beispiel nur Einstellungen für diese Suchdienstanwendung vornehmen.

Delegieren der Verwaltung einer Dienstanwendung

1. Klicken Sie in der Schnellstartleiste der Zentraladministration auf *Anwendungsverwaltung* und dann im Abschnitt *Dienstanwendungen* auf *Dienstanwendungen verwalten*.
2. Klicken Sie auf die Zeile der betreffenden Dienstanwendung.

   Klicken Sie nicht auf den *Namen* einer Dienstanwendung. Die meisten Dienstanwendungsnamen sind Verknüpfungen mit den Verwaltungsprogrammen der Dienstanwendungen.
3. Klicken Sie im Menüband auf *Administratoren*.

Wenn Sie einen Dienstanwendungsadministrator einsetzen, der noch kein Mitglied der Farmadministratorgruppe ist, wird er automatisch zur Gruppe *Delegated Administrators* hinzugefügt. Dadurch ist der Benutzer in der Lage, die Zentraladministrationswebsite zu verwenden.

Dienstanwendungsfeatureadministratoren

Einige Dienstanwendungen haben Features, die sich weiter delegieren lassen. Die *Benutzerprofildienst-Anwendung* unterstützt zum Beispiel das Feature Zielgruppen. Zielgruppen definieren eine Sammlung von Benutzern, die bestimmte Kriterien erfüllen. Der Inhalt kann speziell an diese Zielgruppe angepasst werden, um den Benutzern die Arbeit mit SharePoint zu erleichtern.

Ein Featureadministrator ist für ein oder mehrere Features einer Dienstanwendung zuständig. Diese Administratoren können eine Teilmenge der Dienstanwendungseinstellungen verwalten, aber nicht die gesamte Dienstanwendung. In den nachfolgenden Kapiteln erfahren Sie mehr über SharePoint-Dienstanwendungen, über die Features und über die Delegierung der Featureverwaltung.

Wenn Sie einen Dienstanwendungsfeatureadministrator einsetzen, der noch kein Mitglied der Farmadministratorgruppe ist, wird er automatisch zur Gruppe *Delegated Administrators* hinzugefügt. Dadurch ist der Benutzer in der Lage, die Zentraladministrationswebsite zu verwenden.

Websitesammlungsbesitzer

Jede Websitesammlung kann zwei *Besitzer* haben: den primären Besitzer und den sekundären Besitzer. Wenn Sie mit der Zentraladministration eine Websitesammlung erstellen, können Sie bei der Gelegenheit auch den primären Websitesammlungsadministrator und den sekundären Websitesammlungsadministrator zuweisen. Die Benutzeroberfläche der Zentraladministration lässt zwar vermuten, dass es sich um Websitesammlungsadministratoren handelt, tatsächlich aber sind sie nicht nur Websitesammlungsadministratoren, sondern auch die Besitzer. Sie können keine Websitesammlung ohne primären Besitzer (einen primären Websitesammlungsadministrator) erstellen. In der Zentraladministration können Sie den primären Besitzer auswechseln und einen sekundären Besitzer hinzufügen, ändern oder entfernen.

Zuweisen eines Websitesammlungsbesitzers

1. Klicken Sie in der Schnellstartleiste der Zentraladministration auf *Anwendungsverwaltung*.
2. Klicken Sie im Abschnitt *Websitesammlungen* auf *Websitesammlungsadministratoren ändern*.

Nur Farmadministratoren können mit der Zentraladministration die Websitesammlungsbesitzer ändern.

Websitesammlungsadministratoren

Eine Websitesammlung kann einen oder mehrere Websitesammlungsadministratoren haben. Websitesammlungsadministratoren haben Vollzugriff auf alle Websites einer Websitesammlung. Sie haben Zugriff auf sämtliche Inhalte in allen Websites der Websitesammlung, selbst wenn sie nicht über explizite Berechtigungen für diese Inhalte verfügen. Sie können alle Einstellungen einer Websitesammlung vornehmen.

Zuweisen von Websitesammlungsadministratoren

1. Klicken Sie in der Stammwebsite einer Websitesammlung auf *Websiteaktionen* und dann auf *Websiteeinstellungen*.
2. Klicken Sie im Abschnitt *Benutzer und Berechtigungen* auf *Websitesammlungsadministratoren*.
3. Tragen Sie Benutzer in die Liste der Websitesammlungsadministratoren ein oder entfernen Sie Benutzer. Aufeinanderfolgende Einträge werden jeweils durch ein Semikolon voneinander getrennt.

Jeder Websitesammlungsadministrator kann die Liste der Websitesammlungsadministratoren ändern. Es kann einen oder mehrere Websitesammlungsadministratoren geben. Die primären und sekundären Besitzer der Websitesammlung sind automatisch auch Websitesammlungsadministratoren und es gibt keine Möglichkeit, Besitzer (sie haben Websitesammlungsverwaltungs- und Kontaktrollen) von den Websitesammlungsadministratoren zu trennen. Wenn eine Websitesammlung nur über einen oder zwei Websitesammlungsadministratoren verfügt, sind dieselben Benutzer auch primäre und sekundäre Besitzer der Websitesammlung.

Details zur Bedeutung der Websitesammlungsadministratoren

Die Terminologie und die Benutzeroberfläche von SharePoint haben gerade bei dem Stichwort Websitesammlungsadministrator zu Missverständnissen geführt. Es hilft, wenn man sich die primären und sekundären Websitesammlungsbesitzer als Eigenschaften einer Websitesammlung vorstellt, nämlich als die Attribute `Owner` und `SecondaryContact` eines `SPSite`-Objekts. Diese Attribute sind auch über die Attribute `OwnerLoginName` und `SecondaryContactLoginName` des `SPSiteAdministration`-Objekts der Farmkonfiguration zugänglich. Es gibt nur zwei Besitzer.

Die Bezeichnung Websitesammlungsadministratoren gibt es nur in der Websitesammlung selbst. Wenn ein Benutzer als Websitesammlungsadministrator eingesetzt wird, wird seine Identität in der Benutzerinformationsliste der Websitesammlung mit der `bit2`-Spalte angezeigt. Theoretisch ist jeder Benutzer einer Websitesammlung ein Websitesammlungsadministrator – Sie sind nicht auf zwei beschränkt.

Aus funktionaler Sicht sind die beiden Besitzer Websitesammlungsadministratoren. Besitzer sind aber auch Kontakte. Sie erhalten E-Mail-Benachrichtigungen über Ereignisse wie die anstehende automatische Löschung von unbenutzten Websites und Anfragen nach der Erlaubnis für den Websitezugriff.

Wenn Sie mit der Zentraladministration Besitzer ändern, wird die Liste der Websitesammlungsadministratoren automatisch aktualisiert. Wenn Sie allerdings die Websitesammlungsadministratoren in den Einstellungen der Stammwebsite ändern, werden die Änderungen nicht immer für die Besitzer übernommen. Beispiele:

- Das Hinzufügen eines zweiten Websitesammlungsadministrators zur Website macht den Benutzer nicht zum sekundären Besitzer.
- Den primären Besitzer aus der Liste der Websitesammlungsadministratoren zu entfernen ist nur möglich, wenn ein zweiter Besitzer zugeordnet wurde. Dann wird der sekundäre Besitzer automatisch zum primären Besitzer.
- Wenn Sie die Reihenfolge des ersten und zweiten Websitesammlungsadministrators in der Website vertauschen und diese beiden Benutzer bereits Besitzer sind, werden sie nicht in den Attributen für die primären und sekundären Besitzer vertauscht.
- Wenn Sie den sekundären Besitzer aus der Liste der Websitesammlungsadministratoren entfernen, wird er auch als sekundärer Besitzer entfernt, aber der nachfolgende Websitesammlungsadministrator wird nicht als neuer Besitzer übernommen.
- Wenn Sie einen Benutzer als dritten oder weiteren Websitesammlungsadministrator hinzufügen, diesen Benutzer in der Zentraladministration zum Besitzer machen und ihn dann wieder als Besitzer entfernen, wird der Benutzer automatisch auch aus der Liste der Websitesammlungsadministratoren entfernt. Das ist vielleicht nicht das, was Sie eigentlich wollten. Vermutlich sind Sie davon ausgegangen, dass der Benutzer immer noch über Vollzugriff auf die Websitesammlung verfügt.
- Ein Benutzer, der als Websitesammlungsadministrator eingesetzt wurde, kann sich selbst zum Besitzer machen, wobei er die aktuellen Besitzer aus der Liste der Websitesammlungsadministratoren entfernt. Vielleicht ist dies nach Ihren Steuerungsplänen zulässig, vielleicht aber auch nicht. Dann müssen Sie solche Änderungen überwachen und kontrollieren.

Wie Sie sehen, bezeichnet die Benutzeroberfläche Benutzer einfach nur als Websitesammlungsadministratoren, wo sie zwischen Administratoren und Besitzern unterscheiden sollte. Das Verhalten der Benutzeroberfläche kann zu Missverständnissen und Problemen führen.

Daher lautet die Empfehlung, die primären und sekundären Besitzer nach Ihrem Steuerungsplan in der Zentraladministration einzusetzen und den Vollzugriff auf eine Websitesammlung über die Liste der Websitesammlungsadministratoren in der Stammwebsite zu vergeben. Vergessen Sie aber nicht, dass die Besitzer E-Mail-Benachrichtigungen erhalten und automatisch auch Websitesammlungsadministratoren sind.

Als Alternative bietet es sich an, das SharePoint-Objektmodell zu verwenden und mit einem Werkzeug wie Windows PowerShell zu arbeiten. Im Objektmodell heißen Besitzer tatsächlich Besitzer – oder Kontakte. Wahrscheinlich bleibt die Sache immer etwas unübersichtlich, wie man es auch dreht und wendet.

Schnelltest

1. Welche Unterschiede bestehen zwischen den Websitesammlungsadministratoren, die in der Zentraladministration eingesetzt werden, und den Websitesammlungsadministratoren, die in der Websitesammlung eingesetzt werden?
2. Verfügen beide Sätze von Websitesammlungsadministratoren in einer Websitesammlung über dieselben Berechtigungen?

Antworten zum Schnelltest

1. In der Zentraladministration können nur zwei Websitesammlungsadministratoren eingesetzt werden. Diese beiden Benutzer erhalten E-Mail-Benachrichtigungen, beispielsweise über Kontingente und über die Websitenutzung. In der Websitesammlung können mehr als nur zwei Websitesammlungsadministratoren eingesetzt werden. Diese Benutzer erhalten aber nicht diese E-Mail-Benachrichtigungen.
2. Beide Sätze von Websitesammlungsadministratoren verfügen in der Websitesammlung über Vollzugriff.

Websitegruppen und Berechtigungen

Jede Website verfügt über Benutzergruppen, denen bestimmte Berechtigungen zugewiesen werden. Die Standardgruppe der Websitebesitzer hat zum Beispiel die Berechtigung *Vollzugriff* für die Website. Websiteberechtigungen werden standardmäßig an alle Unterwebsites, Listen, Bibliotheken, Ordner, Elemente und Dokumente vererbt. Daher verfügt die Gruppe *Besitzer* einer Website in der obersten Ebene einer Websitesammlung über Vollzugriff auf den Inhalt – wie auch die Websitesammlungsadministratoren.

Die Gruppe *Besitzer* und jede andere Gruppe mit Vollzugriff auf eine Website kann Verwaltungsarbeiten an der Website und jeder ihrer Listen und Bibliotheken durchführen. Anders als bei Websitesammlungsadministratoren kann die Vollzugriffsberechtigung der Besitzergruppe auf Inhalte durch eine Deaktivierung der Vererbung außer Kraft gesetzt werden. Außerdem kann die Gruppe *Besitzer* zwar einige, aber nicht alle Einstellungen auf Websiteebene ändern, einschließlich der Einstellungen, die für die ganze Websitesammlung gelten.

In Kapitel 4 wird die Verwaltung von Websitegruppen und Berechtigungen ausführlicher behandelt.

Hinweis Weitere Terminologieprobleme

Die SharePoint-Terminologie macht hier offensichtlich eine weitere Wendung zum Schlechteren: Die Besitzergruppe einer Website ist nicht dieselbe wie die Gruppe der Websitesammlungsbesitzer, die in der Benutzeroberfläche der Zentraladministration wiederum Websitesammlungsadministratoren genannt wird, obwohl Websitesammlungsbesitzer E-Mail-Benachrichtigungen erhalten, die Websitesammlungsadministratoren nicht erhalten. Verwirrt? Da ergeht es Ihnen wie jedem anderen auch. Nehmen Sie sich die Zeit, die Rollen und die Gültigkeitsbereiche dieser Rollen zu verstehen, damit Sie unabhängig von dem Begriff, den ein Kollege benutzt oder der in einem Text verwendet wird, interpretieren können, welche Rolle tatsächlich gemeint ist.

Weitere Informationen Administratoren und Besitzer

Der Artikel »Auswählen von Administratoren und Besitzern für die Verwaltungshierarchie (SharePoint Server 2010)« unter *http://go.microsoft.com/fwlink/?LinkID=192722* bietet weitere Informationen über Websitesammlungsbesitzer und Administratoren.

SharePoint-Verwaltungsprogramme

Mit jeder neuen Version von SharePoint wurden neue Verwaltungsprogramme eingeführt. Für SharePoint 2007 hat Microsoft die Zentraladministration überarbeitet und Stsadm (*Stsadm.exe*) eingeführt, das 182 Befehle bietet. Inzwischen gilt Stsadm zwar wieder als veraltet, lässt sich in SharePoint 2010 aber noch verwenden.

Für SharePoint 2010 hat Microsoft wiederum die Zentraladministration überarbeitet und auf der Basis der Aufgaben strukturiert. Außerdem steht für viele Verwaltungsseiten ein Menüband zur Verfügung. Auf der Befehlszeile schließt sich SharePoint 2010 anderen Microsoft-Technologien an, in denen Windows PowerShell als primäres Verwaltungsprogramm verwendet wird. In SharePoint 2010 gibt es über 600 Windows PowerShell-Cmdlets für die Verwaltung einer SharePoint-Farm. Windows PowerShell bietet eine Obermenge der Fähigkeiten der Zentraladministration. Zur Installation von SharePoint ist Windows PowerShell 2.0 erforderlich. Es wird gegebenenfalls vom Vorbereitungstool für Microsoft SharePoint 2010-Produkte (PrerequisiteInstaller) installiert.

Stsadm

Stsadm.exe ist ein Befehlszeilenprogramm, das im Ordner *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\BIN* zu finden ist. Stsadm gilt zwar als veraltet, ist aber aus Gründen der Abwärtskompatibilität zu älteren Produktversionen noch im Produkt vorhanden. Für einige Vorgänge, die mit Stsadm möglich sind, gibt es noch keine Entsprechung in Windows PowerShell.

Einige Vorgänge, die mit Stsadm durchgeführt werden konnten, werden in SharePoint 2010 nicht mehr unterstützt, weil sich Features und die Architektur geändert haben. Beispielsweise gibt es keine Befehle zur Erstellung, Auflistung und Verwaltung von Anbietern für gemeinsame Dienste (SSPs) mehr, weil SSPs durch Dienstanwendungen ersetzt wurden.

Um Stsadm zu verwenden, öffnen Sie auf dem SharePoint-Server eine Eingabeaufforderung mit der Option *Als Administrator ausführen* und navigieren in den Ordner, der die Datei *Stsadm.exe* enthält: *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\BIN*. Diesen Wechsel in einen so tief geschachtelten Ordner können Sie vermeiden, indem Sie den Pfad zu diesem Ordner in die Umgebungsvariable `Path` eintragen, beispielsweise mit folgendem Befehl:

```
set path=%path%;C:\Program Files\Common Files\Microsoft Shared\
    Web Server Extensions\14\BIN
```

Als Alternative können Sie SharePoint 2010-Verwaltungsshell oder Windows PowerShell mit geladenem SharePoint-Snap-In verwenden. Beide fügen den Pfad zum *BIN*-Ordner zur `Path`-Variablen hinzu.

Stsadm bietet seine Funktionen als *Vorgänge* (operations) an. Der Aufruf erfolgt mit folgender Syntax:

```
stsadm -o <VorgangsName> [-parameter <Wert> ...]
```

Darin gilt:

- *<VorgangsName>* ist der Name eines Stsadm-Vorgangs.
- *<Wert>* ist der Wert für einen Parameter, der für den Vorgang verwendet wird.

Mit folgendem Befehl erhalten Sie eine Übersicht über die unterstützten Vorgänge:

```
stsadm -help
```

Wenn Sie die Dokumentation zu einem bestimmten Vorgang und der verwendeten Parameter lesen möchten, geben Sie folgenden Befehl ein:

```
stsadm -help <VorgangsName>
```

Windows PowerShell

Windows PowerShell ist das aktuelle Werkzeug für die Verwaltung und Automatisierung von Microsoft-Technologieplattformen auf der Befehlszeile. SharePoint 2010 bietet ein Snap-In mit über 600 Cmdlets für die Verwaltung von SharePoint mit Windows PowerShell. Dieses Thema wird in Lektion 2 dieses Kapitels ausführlicher behandelt.

SharePoint 2010-Verwaltungsshell

Bei der Installation von SharePoint 2010 wird auch die SharePoint 2010-Verwaltungsshell installiert. Das ist die bevorzugte Umgebung für die Ausführung von aufgabenbezogenen Befehlen und zur Ausführung von Skripts. Die SharePoint 2010-Verwaltungsshell unterstützt Stsadm und Windows PowerShell. Daher handelt es sich nicht um ein separates Verwaltungsprogramm, sondern eher um eine Umgebung, in der sich die beiden wichtigsten Befehlszeilenprogramme verwenden lassen.

In der SharePoint 2010-Verwaltungsshell enthält die `Path`-Variable bereits den Ordner *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\BIN*, in dem die Datei *Stsadm.exe* zu finden ist. Sie können also Stsadm-Vorgänge ausführen, ohne in diesen Ordner wechseln zu müssen. Vergessen Sie aber nicht, die SharePoint 2010-Verwaltungsshell mit der Option *Als Administrator ausführen* zu öffnen, damit sich Stsadm-Vorgänge ausführen lassen.

In der SharePoint 2010-Verwaltungsshell gibt es auch ein Windows PowerShell-Profil. Es sorgt dafür, dass das SharePoint-Snap-In geladen wird, und nimmt einige Einstellungen vor, um die Leistung von Windows PowerShell bei der SharePoint-Verwaltung zu optimieren. Über die SharePoint 2010-Verwaltungsshell erfahren Sie in Lektion 2 mehr.

Zentraladministration

Die Zentraladministration ist die Webanwendung, mit der Sie bei den Übungen in diesem Buch bisher gearbeitet haben, um SharePoint 2010 zu konfigurieren. Wie bei jeder Webanwendung lässt sich auch der Zugriff auf diese Anwendung beschränken. Ob Sie die Zentraladministration verwenden können, hängt von den Sicherheitsberechtigungen ab. Ihr Benutzerkonto, mit dem Sie Verwaltungsarbeiten durchführen, muss Mitglied der Gruppe *Farm Administrators* sein, damit Sie mit der Zentraladministration arbeiten können. In Kapitel 1 wurde bereits beschrieben, dass Sie Ihr eigenes Konto nach der Installation von SharePoint 2010 zur Gruppe *Farm Administrators* hinzufügen sollten, solange Sie noch im Kontext eines Setup-Benutzerkontos (wie *SP_Admin*) arbeiten.

Benutzer, denen Verwaltungsrollen für Dienstanwendungen oder Dienstanwendungsfunktionen übertragen wurden, können ebenfalls die Zentraladministration öffnen. Allerdings wird der Funktionsumfang der Zentraladministration dann aus Sicherheitsgründen eingeschränkt, und sie zeigt nur die Navigationsverknüpfungen und Befehle an, die der Benutzer verwenden darf. Alle anderen Navigationsverknüpfungen und Optionen bleiben verborgen.

Der Sicherheitskontext der Zentraladministration

Nachdem Sie Zugriff auf die Zentraladministration erhalten haben, werden die Arbeiten, die Sie durchführen, nicht im Kontext Ihres eigenen Kontos durchgeführt. Jede Änderung, die Sie in der Zentraladministration vornehmen, wird mit der Anwendungspoolidentität für die Zentraladministration und den Zeitgeberdienst durchgeführt, beispielsweise mit *SP_Farm*.

Wenn etwas nicht funktioniert, sollten Sie überprüfen, ob die Identität *SP_Farm* über die erforderlichen Berechtigungen verfügt. Für einige Arbeiten, die in der Zentraladministration durchgeführt werden, muss das Konto zum Beispiel über folgende Eigenschaften verfügen:

- Mitgliedschaft in der lokalen Gruppe *Administratoren* auf jedem SharePoint-Server. Gewöhnlich sollte *SP_Farm* kein Mitglied der Gruppe *Administratoren* sein, und SharePoint 2010 weist in der Zentraladministration mit einer entsprechenden Meldung darauf hin, wenn es erkennt, dass *SP_Farm* Mitglied der Administratorengruppe ist. Allerdings muss das Konto für die Bereitstellung der Benutzerprofildienst-Anwendung Mitglied der Gruppe *Administratoren* sein.
- Berechtigungen für Microsoft SQL Server. Bei der Konfiguration von SharePoint erhält das Konto *SP_Farm* die Berechtigungen, die es für Zugriffe auf SQL Server und die Datenbanken braucht. Werden diese Berechtigungen aber unter SQL Server entfernt, wirkt sich dies auf die SharePoint-Funktionen aus und die Verwaltungsvorgänge können fehlschlagen.

Ändern des Ports der Zentraladministration

Bei der Konfiguration mit dem *Konfigurations-Assistenten für SharePoint-Produkte* (*Psconfigui.exe*) geben Sie den Port an, an den die Zentraladministration gebunden ist. Diesen Port können Sie später zwar nicht in der Zentraladministration ändern, aber mit Stsadm oder Windows PowerShell.

Ändern des Ports der Zentraladministration mit Stsadm

Mit dem Stsadm-Vorgang `setadminport` können Sie die Zentraladministration an einen anderen Port binden.

```
stsadm -o setadminport -port <PortNummer>
```

Dabei gilt:

- *<PortNummer>* ist ein verfügbarer Port.

Ändern des Ports der Zentraladministration mit Windows PowerShell

Mit dem Cmdlet `Set-SPCentralAdministration` können Sie die Zentraladministration an einen anderen Port binden.

```
Set-SPCentralAdministration -Port <PortNummer>
```

Dabei gilt:

- *<PortNummer>* ist ein verfügbarer Port, dessen Nummer größer als 1023 und kleiner als 32767 ist.

Windows PowerShell ist das Thema der nächsten Lektion.

Weitere Informationen Ändern des Zentraladministrationsports

Die TechNet-Artikel »Ändern der Portnummer der Website für die Zentraladministration« unter *http://go.microsoft.com/fwlink/?LinkID=192720* und »Setadminport: Stsadm-Vorgang« unter *http://go.microsoft.com/fwlink/?LinkID=192721* beschreiben, wie sich der Port der Zentraladministration ändern lässt.

Übung Konfigurieren der SharePoint-Verwaltung

Die Übungen leiten Sie bei wichtigen Arbeitsschritten an. Die Beschreibungen gehen an dieser Stelle allerdings nicht ins Detail. Gehen Sie sorgfältig vor und wenden Sie die Verfahren an, die in dieser Lektion und an anderen Stellen des Buchs beschrieben werden. Eine ausführlichere Anleitung finden Sie bei Bedarf im Abschnitt »Ausführliche Übungsanleitungen« am Ende dieses Buchs.

In dieser Übung delegieren Sie die Verwaltung der Zentraladministration und die Verwaltung von Dienstanwendungen, Features, Websitesammlungen und Websites. Außerdem konfigurieren Sie den Port für die Zentraladministration mit Stsadm.

Vorbereiten der Übungen

Bevor Sie diese Übungen durchführen, müssen Sie eine Testumgebung aufbauen, wie in der Einführung dieses Buchs beschrieben.

1. Wenden Sie den Snapshot *CHAPTER 01* auf CONTOSO-DC an.
2. Wenden Sie den Snapshot *CHAPTER 01* auf SP2010-WFE1 an.
3. Starten Sie CONTOSO-DC.

   Warten Sie, bis der virtuelle Computer hochgefahren ist und die Aufforderung *Drücken Sie Strg+Alt+Entf, um sich anzumelden* erscheint.
4. Starten Sie SP2010-WFE1.

▶ Übung 1 Hinzufügen eines Benutzers zur Gruppe *Farm Administrators*

In dieser Übung fügen Sie ein Benutzerkonto zur Gruppe *Farm Administrators* hinzu.

1. Melden Sie sich als *CONTOSO\SP_Admin* mit dem Kennwort **Pa$$w0rd** auf SP2010-WFE1 an.
2. Fügen Sie den Benutzer Pat Coleman zur Gruppe *Farm Administrators* hinzu. Sein Anmeldename ist *PatC*.

▶ Übung 2 Anmelden als anderer Benutzer

In dieser Übung melden Sie sich mit dem Konto an, das Sie gerade zur Gruppe *Farm Administrators* hinzugefügt haben.

- Melden Sie sich als *CONTOSO\PatC* mit dem Kennwort **Pa$$w0rd** bei der Zentraladministration an.

▶ Übung 3 Zuweisen eines Websitesammlungsbesitzers

In dieser Übung fügen Sie einen Websitesammlungsbesitzer für die Websitesammlung *Contoso Intranet* hinzu.

- Tragen Sie Pat Coleman in der Zentraladministration als sekundären Websitesammlungsadministrator für die Intranet-Websitesammlung ein.

▶ Übung 4 Zuweisen eines Websitesammlungsadministrators

In dieser Übung fügen Sie einen Websitesammlungsadministrator für die Websitesammlung *Contoso Intranet* hinzu.

1. Öffnen Sie im Internet Explorer eine neue Registerkarte und navigieren Sie zu *http://intranet.contoso.com*.
2. Tragen Sie April Meyer als Websitesammlungsadministrator ein. Ihr Benutzername ist *AprilM*.

▶ Übung 5 Zuweisen eines Websitebesitzers

In dieser Übung tragen Sie einen Besitzer für die Contoso-Intranetwebsite ein.

- Fügen Sie Kevin Cook zur Gruppe der Contoso-Intranetbesitzer hinzu. Sein Benutzername ist *KevinC*.

▶ Übung 6 Zuweisen eines Dienstanwendungsadministrators

In dieser Übung delegieren Sie die Verwaltung der Dienstanwendung *Verwalteter Metadatendienst*.

- Übertragen Sie April Meyer in der Zentraladministration die Verwaltung des Dienstes für verwaltete Metadaten. Geben Sie Ihr Vollzugriff.

▶ Übung 7 Ändern des Ports der Zentraladministration

In dieser Übung ändern Sie den Port für die Zentraladministration.

1. Öffnen Sie eine Eingabeaufforderung mit der Option *Als Administrator ausführen*.
2. Geben Sie folgenden Befehl ein und drücken Sie dann die EINGABETASTE:

   ```
   stsadm -o setadminport -port 9998
   ```

   Eine Fehlermeldung wird angezeigt.
3. Bestimmen Sie die Ursache des Fehlers und geben Sie den Befehl korrekt ein, damit er ausgeführt werden kann.
4. Navigieren Sie im Internet Explorer zu *http://sp2010-wfe1:9998*.

   Die Zentraladministration öffnet sich. Da die Webanwendung neu kompiliert und zwischengespeichert werden muss, dauert dies einige Zeit.
5. Öffnen Sie eine Eingabeaufforderung und ändern Sie den Port für die Zentraladministration wieder auf Port 9999.
6. Navigieren Sie im Internet Explorer zu *http://sp2010-wfe1:9999*.

   Die Zentraladministration öffnet sich. Da die Webanwendung neu kompiliert und zwischengespeichert werden muss, dauert dies einige Zeit.

Zusammenfassung der Lektion

- Zu den SharePoint-Verwaltungsrollen gehören die SharePoint-Gruppe *Farm Administrators*, die Gruppe *Administratoren* auf dem SharePoint-Server, Dienstanwendungs- und Featureadministratoren, die primären und sekundären Besitzer von Websitesammlungen, Websitesammlungsadministratoren und Websitegruppen mit Verwaltungsberechtigungen.
- Farmadministratoren können in der Zentraladministration die primären und sekundären Besitzer einer Websitesammlung ändern. Primäre und sekundäre Besitzer sind automatisch auch Websitesammlungsadministratoren. Außerdem erhalten sie E-Mail-Benachrichtigungen über die Websitenutzung, Websitelöschungen und Anfragen nach Zugriff auf die Website.
- Websitesammlungsadministratoren können in den Websiteeinstellungen für die Stammwebsite einer Websitesammlung die Liste der Websitesammlungsadministratoren ändern. Bestimmte Änderungen in dieser Liste führen zu Änderungen bei den primären und sekundären Besitzern der Websitesammlung.
- SharePoint 2010 bietet zur Verwaltung die Zentraladministration, Stsadm und Windows PowerShell. Die SharePoint 2010-Verwaltungsshell unterstützt *Stsadm.exe* und SharePoint-Cmdlets für Windows PowerShell.
- *Stsadm.exe* ist zwar in SharePoint 2010 noch verfügbar, aber das bevorzugte Werkzeug für die Verwaltung und Automatisierung auf der Befehlszeile ist Windows PowerShell.
- Mitglieder der Gruppe *Farm Administrators* sowie Dienstanwendungs- und Featureadministratoren erhalten Zugriff auf die Zentraladministration.
- In der Zentraladministration durchgeführte Arbeiten erfolgen im Sicherheitskontext der Zentraladministrations-Anwendungspoolidentität, beispielsweise *SP_Farm*.

Lernzielkontrolle

Mit den folgenden Fragen können Sie Ihr Wissen über den Stoff aus Lektion 1, »SharePoint-Verwaltungsrollen und Werkzeuge«, überprüfen. Die Fragen finden Sie (in englischer Sprache) auch auf der Begleit-CD, Sie können sie also auch auf dem Computer im Rahmen eines Übungstests beantworten.

Hinweis Die Antworten

Die Antworten auf diese Fragen mit Erklärungen, warum die jeweiligen Auswahlmöglichkeiten richtig oder falsch sind, finden Sie im Abschnitt »Antworten« am Ende dieses Buchs.

1. Sie möchten fünf Benutzern die Möglichkeit geben, in der Websitesammlung der Abteilung *Sales* Features zu aktivieren und zu deaktivieren. Welche der folgenden Elemente konfigurieren Sie?
   A. Websitesammlungsadministratoren in der Zentraladministration
   B. Einstellungen für Websitesammlungsadministratoren in der Stammwebsite der Abteilung
   C. Besitzer der Sales-Websitesammlung
   D. Mitwirkende der Sales-Websitesammlung

2. Sie möchten sicherstellen, dass Lola Jacobsen E-Mail-Benachrichtigungen erhält, wenn die Websitesammlung ihrer Abteilung ihr zugeteiltes Kontingent ausgeschöpft hat. Wie erreichen Sie das? (Wählen Sie alle zutreffenden Antworten. Jede korrekte Antwort ist Teil der vollständigen Lösung.)
   A. Sie fügen Lola über die Seite *Websiteeinstellungen* der Website als drittes Mitglied zur Gruppe der Websitesammlungsadministratoren hinzu.
   B. Sie setzen Lola in der Zentraladministration als primären Websitesammlungsadministrator ein.
   C. Sie richten eine Kontingentvorlage namens *Sales Quota* mit einer Speicherbegrenzung ein.
   D. Sie richten eine Kontingentvorlage namens *Sales Quota* mit einer Warnstufe ein.
   E. Sie wenden eine Kontingentvorlage namens *Sales Quota* auf die Websitesammlung *Sales* an.
   F. Sie wählen auf der Seite *Kontingente und Sperren für Websitesammlungen* die Vorlage *Persönliches Kontingent*.
   G. Sie konfigurieren ausgehende E-Mails für die Farm.
   H. Sie konfigurieren eingehende E-Mails für die Farm.
3. Sie erhalten die Fehlermeldung *Zugriff verweigert*, wenn Sie versuchen, mit *Stsadm.exe* den primären Besitzer einer Websitesammlung festzulegen. Derzeit verfügt die Websitesammlung nur über einen Besitzer. Wie können Sie das Problem beheben?
   A. Sie können mit Stsadm keinen primären Besitzer für eine Websitesammlung festlegen.
   B. Sie starten Stsadm mit der Option *Als Administrator ausführen*.
   C. Sie fügen den Pfad zu Stsadm zur Umgebungsvariablen `Path` hinzu.
   D. Sie entfernen den aktuellen primären Websitesammlungsadministrator.

Lektion 2: Automatisieren von SharePoint-Vorgängen mit Windows PowerShell

Diese Lektion bietet Ihnen eine Einführung in Windows PowerShell, das empfohlene Tool für die Verwaltung und Automatisierung von SharePoint 2010 auf der Befehlszeile. Windows PowerShell hat das »Power« im Namen sicherlich verdient. Wie Sie noch feststellen werden, ermöglicht es die Durchführung von Arbeiten, die in der Zentraladministration viele Mausklicks erfordern, auf einer einzigen Befehlszeile, und was noch besser ist, es erleichtert auch die Arbeiten, die wiederholt durchgeführt werden müssen.

Am Ende dieser Lektion werden Sie in der Lage sein, die folgenden Aufgaben auszuführen:

- Beschreiben der Rolle von Windows PowerShell für die Verwaltung von SharePoint
- Beschreiben der SharePoint 2010-Verwaltungsshell
- Beschreiben der logischen Struktur von SharePoint
- Verwenden von Windows PowerShell zur Suche nach Cmdlets und zur Anzeige der integrierten Dokumentation
- Unterscheiden zwischen der Terminologie, die in der Benutzeroberfläche und im Objektmodell für SharePoint-Komponenten verwendet wird
- Beschreiben der Objekte, Member, Eigenschaften und Methoden von Windows PowerShell
- Beschreiben, wie Ausgaben in Windows PowerShell ausgewählt, sortiert und formatiert werden
- Beschreiben, wie Objekte gefiltert werden
- Erstellen eines SharePoint-Intranets mit Windows PowerShell
- Erläutern der Konzepte von Iteration und Variablen
- Automatisieren von SharePoint-Vorgängen mit Windows PowerShell
- Delegieren der Berechtigung zur Verwendung von Windows PowerShell
- Ausführen von Windows PowerShell-Skripts

Veranschlagte Zeit für diese Lektion: 90 Minuten

Prüfungstipp

Dieser Abschnitt soll Ihnen nur eine Einführung in Windows PowerShell bieten, damit Sie mit diesem wichtigen Verwaltungstool vertraut werden. In der Prüfung 70-667 wird nicht von Ihnen erwartet, Windows PowerShell-Skripts zu erstellen. Allerdings sollten Sie Cmdlets erkennen, die wie die in diesem Buch beschriebenen Cmdlets für Verwaltungsarbeiten in SharePoint verwendet werden. Weitere Informationen über die Erstellung von Automatisierungsskripts für Windows PowerShell finden Sie im *Windows PowerShell 2.0 Administrator's Pocket Consultant* von William R. Stanek (Microsoft Press, 2009).

Hinweis Eine Führung durch Windows PowerShell

Diese Lektion ist eine Führung durch Windows PowerShell. Am Ende der Lektion vertiefen Sie den Stoff der Lektion durch einige Übungen. Wenn Sie die Einarbeitung in Windows PowerShell noch etwas fortsetzen möchten, finden Sie in den empfohlenen Übungen am Ende des Kapitels einige Anregungen.

Einführung in Windows PowerShell

Windows PowerShell ist eine aufgabenorientierte Befehlszeilenshell und Skriptsprache, die speziell für die Systemverwaltung entwickelt wurde. Aufbauend auf dem Microsoft .NET Framework unterstützt Windows PowerShell IT-Fachleute bei der Steuerung und der Automatisierung der Verwaltung von einigen Microsoft-Technologien, wie dem Betriebssystem Windows, SharePoint 2010, dem Active Directory-Domänendienst und Microsoft Exchange Server.

Mit den Windows PowerShell-Befehlen, die *Cmdlets* genannt werden, können Sie Verwaltungsarbeiten auf der Befehlszeile durchführen. Mit Windows PowerShell-*Anbietern* können Sie genauso einfach auf Datenspeicher wie Registrierung und Zertifikatspeicher zugreifen wie auf das Dateisystem. Außerdem verfügt Windows PowerShell über einen leistungsfähigen Parser für Ausdrücke und eine ausgereifte Skriptsprache.

Windows PowerShell weist folgende Leistungsmerkmale auf:

- Cmdlets zur Durchführung von Systemverwaltungsaufgaben
- Eine aufgabenorientierte Skriptsprache
- Unterstützung für vorhandene Skripts und Befehlszeilentools. Sie können zum Beispiel die meisten *Cmd.exe*-Befehle in Windows PowerShell ausführen.
- Einheitliches Design. Da Cmdlets und Systemdatenspeicher eine einheitliche Syntax und Namenskonvention verwenden, können Daten leicht ausgetauscht werden. Die Ausgabe eines Cmdlets kann beispielsweise ohne Neuformatierung oder Überarbeitung als Eingabe für ein anderes Cmdlet verwendet werden.
- Systemressourcen wie die Registrierung, der Zertifikatspeicher und der Verzeichnisdienst werden durch Anbieter zugänglich gemacht und ermöglichen eine Navigation mit derselben Technik, mit der Benutzer im Dateisystem navigieren.
- Leistungsfähige Mittel zur Objektbearbeitung. Sie können Objekte selbst ändern oder sie an andere Tools oder Datenbanken senden.
- Erweiterbare Schnittstelle. Unabhängige Softwareanbieter und Entwickler können benutzerdefinierte Tools und Hilfsprogramme für die Verwaltung ihrer Software erstellen.

Stsadm und Windows PowerShell überlappen sich in ihrer Unterstützung der Vorgänge, die in SharePoint 2007 und SharePoint 2010 gebräuchlich sind. Allerdings unterstützt nur Windows PowerShell die Verwaltung der neuen Features und Aufgaben wie die folgenden:

- Installation und Konfiguration von SharePoint 2010
- Verwalten von Dienstanwendungen
- Fein abgestufte Steuerung beim Sichern und Wiederherstellen

Eine der wichtigsten neuen Funktionen von Windows PowerShell 2.0 ist dessen Fähigkeit zur Fernsteuerung anderer Computer (remoting). Sie können Windows PowerShell-Befehle auf Remotesystemen ausführen. Da es sich bei dieser Fernsteuerung um eine Fähigkeit von Windows PowerShell handelt und nicht speziell um eine Funktion von SharePoint, wird sie in diesem Buch nicht weiter behandelt. Der TechNet-Artikel »Ausführen von Remotebefehlen« unter *http://go.microsoft.com/fwlink/?LinkID=192745* beschreibt die Fernsteuerung mit Windows PowerShell. Vielleicht interessiert Sie auch der Blogeintrag »SharePoint 2010 with Windows PowerShell Remoting Step by Step« unter *http://blogs.msdn.com/b/opal/archive/2010/03/07/sharepoint-2010-with-windows-powershell-remoting-step-by-step.aspx*.

SharePoint 2010-Verwaltungsshell

Sie können SharePoint auf zwei Arten mit Windows PowerShell verwalten, nämlich in der Windows PowerShell-Konsole und in der SharePoint 2010-Verwaltungsshell. Wie in Lektion 1 erwähnt, ist die SharePoint 2010-Verwaltungsshell das bevorzugte Tool für die Verwaltung von SharePoint auf der Befehlszeile, weil es die Verwendung von Windows PowerShell und von *Stsadm.exe* unterstützt.

Öffnen der SharePoint 2010-Verwaltungsshell

So öffnen Sie die SharePoint 2010-Verwaltungsshell:

- Klicken Sie nacheinander auf *Start*, *Alle Programme*, *Microsoft SharePoint 2010 Products* und dann auf *SharePoint 2010-Verwaltungsshell*.

Für Stsadm und einige Windows PowerShell-Cmdlets müssen Sie die SharePoint 2010-Verwaltungsshell allerdings mit erhöhten Rechten öffnen. Diese Cmdlets und Stsadm lassen sich sonst nicht in der SharePoint 2010-Verwaltungsshell verwenden.

Öffnen der SharePoint 2010-Verwaltungsshell mit erhöhten Rechten

So öffnen Sie die SharePoint 2010-Verwaltungsshell mit erhöhten Rechten:

1. Klicken Sie auf *Start*, *Alle Programme*, *Microsoft SharePoint 2010 Products*, klicken Sie *SharePoint 2010-Verwaltungsshell* mit der rechten Maustaste an und klicken Sie dann auf *Als Administrator ausführen*.

   Das Dialogfeld *Benutzerkontensteuerung* öffnet sich.
2. Klicken Sie auf *Ja*.

Cmdlets

In herkömmlichen Shells wie der Eingabeaufforderung (*Cmd.exe*) verwenden Sie Befehle wie `Dir` oder `Copy`, die in die Shell integriert sind, oder Sie rufen ausführbare Programme wie *Attrib.exe* oder *Xcopy.exe* auf, die gewöhnlich zusätzliche Parameter von der Befehlszeile einlesen und Rückmeldungen in Form von Textausgaben, Fehlermeldungen und Fehlercodes liefern.

In Windows PowerShell verwenden Sie Cmdlets. Ein Cmdlet ist ein spezialisierter Befehl mit einer bestimmten Aufgabe, der auf ein Objekt wirkt. SharePoint Server 2010 wird mit mehr als 500 Cmdlets für Windows PowerShell geliefert. Sie brauchen sie nicht alle auswendig zu lernen. Stattdessen sollten Sie wissen, wie man ein Cmdlet findet, wenn man es braucht, und die dazugehörige Dokumentation anzeigt. Im Lauf der Zeit werden Sie die Cmdlets im Gedächtnis behalten, die Sie häufig verwenden.

Windows PowerShell ist ein modernes Befehlszeilen- und Automatisierungstool, das von den Erfahrungen mit älteren Befehlszeilenumgebungen wie der Eingabeaufforderung profitiert. Schon bei der Einarbeitung ist von Nutzen, dass man Cmdlets auf einfache Weise suchen und deren Syntax leicht lernen kann.

Das Cmdlet `Get-Command` listet Cmdlets auf. Geben Sie einfach folgenden Befehl ein, um alle Cmdlets aufzulisten, die in der Windows PowerShell-Sitzung zur Verfügung stehen:

```
Get-Command
```

Zwischen Groß- und Kleinbuchstaben wird nicht unterschieden. Daher sind zum Beispiel folgende Schreibweisen zulässig:

- `Get-Command`
- `get-command`
- `GET-COMMAND`

Cmdlets werden nach dem Schema Verb-Substantiv (*Verb-Noun*) benannt, das auch als Aktion-Objekt-Format (*Action-Object*) bezeichnet wird. Das Substantiv steht immer im Singular. Zum Beispiel heißt das Cmdlet, das alle auf einem Computer verfügbaren Dienste auflistet, `Get-Service`. Um alle auf einem Computer verfügbaren Dienste aufzulisten, geben Sie folgenden Befehl ein:

```
Get-Service
```

Mit dem Cmdlet `Get-Verb` lässt sich die relativ kleine Menge an Verben auflisten. Die Substantive orientieren sich an den Namensstandards des Windows PowerShell-Teams. Beispielsweise beginnen die Substantive für SharePoint-Cmdlets mit SP.

Auflisten aller SharePoint-Cmdlets

Um alle SharePoint-Cmdlets aufzulisten, geben Sie folgenden Befehl ein:

```
Get-Command -noun SP* | more
```

Windows PowerShell unterstützt weitgehend die Syntax, die in der Eingabeaufforderung üblich ist, was den Umstieg auf Windows PowerShell erleichtert. Wie in der Eingabeaufforderung lässt sich die Textausgabe eines Befehls durch das Hinzufügen von `| more` zu einem Befehl seitenweise anzeigen.

Der hier gezeigte Befehl ist eine Kurzform, die ausnutzt, dass alle SharePoint-Substantive mit SP beginnen. Eine technisch präzisere Lösung ist, alle Befehle aufzulisten, die sich im SharePoint-Snap-In für Windows PowerShell befinden. Windows PowerShell ist erweiterbar und zusätzliche Befehle können mit Snap-Ins hinzugefügt werden. Bei der Installation von SharePoint 2010 wird auch das SharePoint-Snap-In für Windows PowerShell installiert. So listen Sie die Befehle aus dem Snap-In auf:

```
Get-Command -pssnapin Microsoft.SharePoint.PowerShell
```

Get-Help

Nachdem Sie ein Cmdlet gefunden haben, das sich für die durchzuführende Arbeit eignen könnte, können Sie die dazugehörige Dokumentation mit dem Cmdlet `Get-Help` anzeigen. Die einfachste Art des Aufrufs besteht darin, auf der Befehlszeile `Get-Help` einzugeben, gefolgt vom Namen des Cmdlets, über das Sie mehr erfahren möchten. Ein Beispiel:

```
Get-Help Get-Service
```

Ohne zusätzliche Parameter zeigt das Cmdlet `Get-Help` eine Übersicht, eine etwas ausführlichere Beschreibung und die Syntax des Cmdlets. Mit den folgenden optionalen Parametern von `Get-Help` lassen sich Art und Umfang der Ausgabe steuern:

- **-examples** Zeigt Beispiele für die Verwendung des Cmdlets.
- **-detailed** Zeigt ausführliche Informationen über das Cmdlet und seine Parameter. Zeigt außerdem Beispiele.
- **-full** Zeigt die gesamte Dokumentation des Cmdlets.

Um beispielsweise den Hilfetext für das Cmdlet `New-SPContentDatabase` anzuzeigen, einschließlich der Beispiele, geben Sie Folgendes ein:

```
Get-Help New-SPContentDatabase -detailed
```

Wenn Sie Informationen über Cmdlets suchen, fangen Sie am besten mit dem Cmdlet `Get-Help` an, besonders dann, wenn Sie sich gerade in Windows PowerShell einarbeiten. Die Windows PowerShell-Cmdlets sind alle in einem Standardformat gut dokumentiert und die Dokumentation lässt sich mit dem Cmdlet `Get-Help` und den Parametern `-examples`, `-detailed` oder `-full` anzeigen.

Weitere Informationen SharePoint-Cmdlets

Der folgende Artikel bietet weitere Informationen über die Grundlagen der Windows PowerShell-Cmdlets: »Grundlegende Informationen über Windows PowerShell« unter *http://technet.microsoft.com/de-de/library/dd347730.aspx*.

Objekte

Anders als die Eingabeaufforderung, in der Befehle ihre Ergebnisse als Text liefern, der entsprechend ausgewertet und bearbeitet werden muss, geben Windows PowerShell-Cmdlets *Objekte* zurück – Darstellungen der betreffenden Komponenten.

Ein Objekt ist eine spezielle Datenstruktur für die Programmierung. Aus technischer Sicht ist ein .NET-Objekt eine Instanz einer .NET-Klasse, in der die verwendeten Daten und die zulässigen Vorgänge beschrieben werden. Stellen Sie sich ein Objekt als eine für die Verwendung im Computer geeignete Darstellung einer Ressource vor. Wenn Sie in Windows PowerShell zum Beispiel das Cmdlet `Get-Service` verwenden, gibt das Cmdlet ein oder mehrere Objekte zurück, die Dienste darstellen.

Objekte können *Eigenschaften* (properties) haben, auch *Attribute* (attributes) genannt, mit denen Daten der Ressource beschrieben werden. Ein Objekt, das einen Dienst darstellt, verfügt zum Beispiel über Eigenschaften, mit denen der Dienstname und der Startzeitpunkt angegeben werden. Indem Sie den Wert einer Eigenschaft abrufen, rufen Sie gegebenenfalls die entsprechenden Daten aus der Ressource ab. Indem Sie den Wert einer Eigenschaft festlegen, schreiben Sie Daten in die Ressource.

Objekte haben auch *Methoden* (methods), mit denen die Aktionen beschrieben werden, die ein Objekt durchführen kann. Ein Dienstobjekt verfügt zum Beispiel über die Methoden `start` und `stop`. Wenn Sie die Methode eines Objekts aufrufen, das eine Ressource darstellt, führen Sie die Aktion letztlich mit der Ressource selbst durch.

Das Cmdlet `Get-Service` gibt zum Beispiel Objekte zurück, die auf dem Computer verfügbare Dienste darstellen. Geben Sie folgenden Befehl ein, um eine Liste aller auf dem Computer verfügbaren Dienste abzurufen:

```
Get-Service
```

Um die Liste der Dienste einzuschränken, verwenden Sie `Get-Service` mit einem passenden Parameter. Der Parameter `-Name` beschränkt die zurückgegebenen Dienste zum Beispiel auf der Basis ihrer Namen. Der folgende Befehl ruft alle auf einem Computer verfügbaren Dienste ab, deren Namen mit SP beginnen. Viele dieser Dienste haben mit SharePoint zu tun:

```
Get-Service -Name SP*
```

Da `-Name` der Standardparameter des Cmdlets `Get-Service` ist, brauchen Sie den Namen des Parameters nicht explizit anzugeben:

```
Get-Service SP*
```

Windows PowerShell-Cmdlets geben keine Befehle oder Parameter an andere Tools oder Programme weiter, wie es in einer Eingabeaufforderung üblich ist. Stattdessen arbeiten Windows PowerShell-Cmdlets direkt mit den .NET-Objekten. Das Cmdlet `Get-Service` gibt eine *Auflistung* (oder Sammlung, collection) von Objekten zurück, ein Objekt für jeden Dienst auf dem Computer. Das Ergebnis des Cmdlet-Aufrufs wird als Tabelle dargestellt, in der einige Eigenschaften der Dienste aufgeführt sind: Status, Name und Anzeigename (Abbildung 2.1).

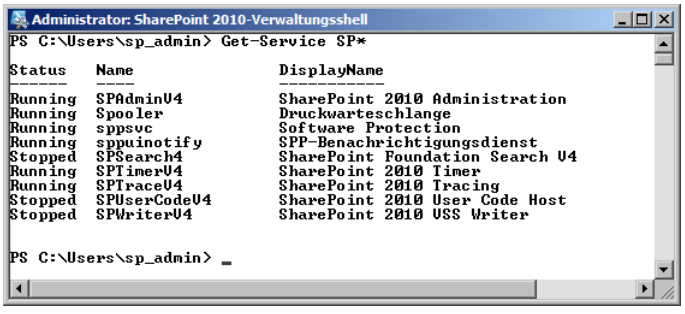

Abbildung 2.1 Das Cmdlet `Get-Service`

Pipeline

In den bisher gezeigten Beispielen wurden nur Eigenschaften von Objekten angezeigt. Allerdings können die Objekte, die ein Cmdlet zurückgibt, auch zur späteren Verwendung in Variablen gespeichert oder als Eingabe an ein anderes Cmdlet weitergegeben werden. Windows PowerShell bietet eine *Pipeline* – einen Kanal, durch den die Ausgabe eines Cmdlets an das folgende Cmdlet weitergegeben werden kann. Die Pipeline wird durch ein Pipe-Symbol (ein senkrechter Strich wie |) dargestellt.

Mit dem folgenden Befehl können Sie zum Beispiel alle Dienste beenden, die auf einem Computer ausgeführt werden und deren Name *SPTimerV4* lautet:

```
Get-Service SPTimerV4 | Stop-Service
```

Das Cmdlet `Get-Service` ruft ein Objekt ab, das den SharePoint-Zeitgeberdienst *SPTimerV4* darstellt, und gibt das Objekt in der Pipeline an das Cmdlet `Stop-Service` weiter, das den Dienst beendet.

Wie bereits erwähnt, besteht einer der wichtigsten Unterschiede zwischen Windows PowerShell und der Eingabeaufforderung darin, dass Cmdlets keinen Text zurückgeben, sondern Objekte. In der Eingabeaufforderung geben die Befehle Text zurück. Dieser Text kann an einen anderen Befehl weitergegeben werden. In Windows PowerShell geben Cmdlets Objekte zurück, die in der Pipeline auf wesentlich leistungsfähigere Weise weiter bearbeitet werden können. Das Cmdlet `Get-Service` gibt im obigen Beispiel ein Objekt zurück, das den *SPTimerV4*-Dienst darstellt. Der nachfolgende Befehl in der Pipeline beendet den Dienst. An dieser Stelle könnte auch ein Cmdlet stehen, das Eigenschaften des Dienstes ändert oder bestimmte Informationen über den Dienst liefert.

Wenn ein Cmdlet mehrere Objekte zurückgibt – eine *Auflistung* von Objekten – und diese Auflistung (oder Sammlung) in der Pipeline weiterreicht, kann ein nachfolgendes Cmdlet jedes dieser Objekte verwenden. Um beispielsweise alle Dienste zu beenden, in deren Anzeigename der Begriff *SharePoint* vorkommt, geben Sie folgenden Befehl ein:

```
Get-Service -DisplayName *SharePoint* | Stop-Service
```

Das Cmdlet `Get-Service` gibt ungefähr zehn Dienstobjekte zurück und reicht diese Auflistung an das Cmdlet `Stop-Service` weiter. Das Cmdlet `Stop-Service` durchläuft diese Objektauflistung und hält jeden Dienst an.

Erweitern der Pipeline auf mehrere Zeilen

Um eine Aufgabe in Windows PowerShell durchzuführen, kann es erforderlich werden, mehrere Cmdlets, Parameter und Ausdrücke zu verwenden. Für kompliziertere Aufgaben ist vielleicht eine lange Befehlszeile mit verschiedenen Elementen erforderlich, wie Funktionen, Schleifen und bedingte Verzweigungen. Häufig wird eine Befehlszeile auch auf mehrere Zeilen umbrochen, um die Lesbarkeit zu verbessern. Man kann auf mehrere Arten angeben, dass die Pipeline auf der nächsten Zeile fortgesetzt wird:

- **Das Tick-Symbol (`)** Wenn ein Tick-Symbol (tick mark, deutsch auch Gravis genannt) das letzte Zeichen einer Zeile ist, dient es als Zeilentrenner und als Fortsetzungszeichen. Windows PowerShell geht dann davon aus, dass es sich bei der nachfolgenden Zeile um eine Fortsetzung der aktuellen Zeile handelt. Im folgenden zweizeiligen Befehl wird ein Tick-Symbol verwendet, um die Lesbarkeit zu verbessern:

  ```
  Get-Service -DisplayName *SharePoint* | `
  Stop-Service
  ```

- **Das Pipe-Symbol (|)** Wenn das Pipe-Symbol das letzte Zeichen auf einer Zeile ist, bedeutet dies ebenfalls, dass die Befehlseingabe noch nicht beendet ist, und Windows PowerShell interpretiert die nachfolgende Zeile als Fortsetzung, wie im folgenden Beispiel:

  ```
  Get-Service -DisplayName *SharePoint* |
  Stop-Service
  ```

- **Eine linke geschweifte Klammer ({)** Geschweifte Klammern schließen andere Strukturen ein, beispielsweise einen Ausdruck oder eine Prozedur. Eine linke geschweifte Klammer bedeutet, dass diese Struktur noch folgt. Im Verlauf dieser Lektion werden Sie noch Beispiele dafür sehen.

Wenn Sie in der Windows PowerShell-Konsole eine Zeile eingeben und diese Zeile mit einem dieser Zeichen endet, zeigt die Konsole als Eingabeaufforderung einen doppelten, nach rechts weisenden spitzen Winkel an, wie in Abbildung 2.2, als sichtbaren Hinweis darauf, dass die Befehlseingabe noch nicht abgeschlossen ist. Um die Eingabe abzuschließen, geben Sie hinter der Eingabeaufforderung einfach nur ein Leerzeichen ein. Dann beginnt Windows PowerShell mit der Ausführung des auf mehreren Zeilen eingegebenen Befehls.

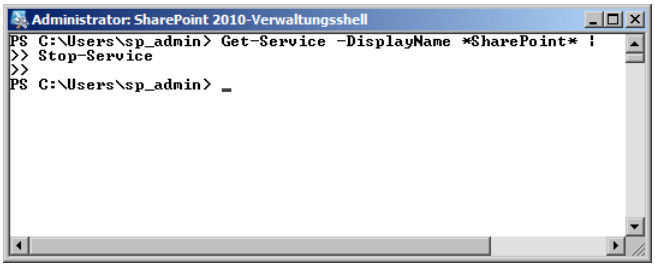

Abbildung 2.2 Ein Befehl kann auf mehreren Zeilen eingegeben werden

Weitere Informationen Die Windows PowerShell-Pipeline

Im Artikel »Grundlegendes zur Windows PowerShell-Pipeline« unter *http://go.microsoft.com/fwlink/?LinkID=192732* finden Sie weitere Informationen über die Windows PowerShell-Pipeline.

Aliasse

In Windows PowerShell kann für ein Cmdlet ein *Alias* verwendet werden, also ein zweiter Name für das Cmdlet. So ist zum Beispiel `gsv` ein anderer Name für `Get-Service`. Das Cmdlet `Get-Alias` listet Aliasse auf. Ohne Parameter listet `Get-Alias` alle Aliasse auf, die in der aktuellen Windows PowerShell-Sitzung definiert sind. Zur Anzeige des Alias für ein bestimmtes Cmdlet verwenden Sie folgenden Befehl:

```
Get-Alias -definition <Cmdlet>
```

Darin ist *<Cmdlet>* der Name des Cmdlets, dessen Aliasse Sie erfahren möchten.

Wenn Sie auf ein Cmdlet stoßen, das sich nicht an die Form Verb-Substantiv hält, handelt es sich um einen Alias. Gelegentlich ist es nicht leicht zu verstehen, was ein Befehl eigentlich bewirkt, wenn ein Alias verwendet wird. Um herauszufinden, welches Cmdlet sich hinter einem Alias verbirgt, geben Sie folgenden Befehl ein:

```
Get-Alias <Alias>
```

Darin ist *<Alias>* der Alias, den Sie überprüfen möchten.

Windows PowerShell-Aliasse ermöglichen die Verwendung von Befehlen, die unter UNIX oder in der Eingabeaufforderung gebräuchlich sind. Beispielsweise lassen sich die Objekte in einem Ordner mit `Dir` und `Ls` auflisten, das sind Aliasse für das Cmdlet `Get-ChildItem`. Die Windows PowerShell-Konsole können Sie mit dem Cmdlet `Clear-Host` löschen oder mit dem Alias `Cls`.

Weitere Informationen Verwenden vertrauter Befehlsnamen

Unter *http://go.microsoft.com/fwlink/?LinkID=192733* finden Sie im Artikel »Verwenden vertrauter Befehlsnamen« weitere Informationen über Windows PowerShell-Aliasse.

Vervollständigung mit der Tabulatortaste

Windows PowerShell unterstützt eine Vervollständigung unvollständig eingegebener Begriffe mit der TABULATORTASTE. Sie geben also einige Buchstaben ein und drücken dann die TABULATORTASTE, um die Eingabe zu vervollständigen. Das gilt nicht nur für Pfade, wie es in der Eingabeaufforderung möglich ist, sondern auch für Cmdlets und ihre Parameter.

Wenn Sie mit der Vervollständigung experimentieren möchten, arbeiten Sie in der SharePoint 2010-Verwaltungsshell die folgende Anleitung zur Erstellung einer neuen Inhaltsdatenbank für eine Webanwendung durch:

1. Geben Sie **New-SPCont** ein und drücken Sie die TABULATORTASTE.

   Windows PowerShell vervollständigt den Namen des Cmdlets zu `New-SPContentDatabase`.

   Der erste Parameter für das Cmdlet `New-SPContentDatabase` ist der Name der Datenbank, die erstellt werden soll.
2. Drücken Sie auf die LEERTASTE, geben Sie **TestContentDB** ein und drücken Sie wieder die LEERTASTE.

   Der nächste Parameter ist der Name des Datenbankservers, auf dem die Inhaltsdatenbank erstellt werden soll.
3. Geben Sie **-Da** ein und drücken Sie die TABULATORTASTE.

   Windows PowerShell vervollständigt den Parameternamen `-DatabaseServer`.
4. Drücken Sie die LEERTASTE, geben Sie **SP2010-WFE1** ein und drücken Sie wieder die LEERTASTE.

   Der nächste erforderliche Parameter ist der Name der Webanwendung, der die Inhaltsdatenbank zugeordnet werden soll.
5. Geben Sie **-W** ein und drücken Sie die TABULATORTASTE.

   Windows PowerShell vervollständigt den Parameternamen `-WebApplication`.
6. Drücken Sie die LEERTASTE und geben Sie **"http://intranet.contoso.com"** ein.
7. Drücken Sie STRG+C, um den Befehl abzubrechen, ohne ihn auszuführen.

Untersuchen und Dokumentieren der logischen Struktur von SharePoint mit Windows PowerShell

Nachdem Sie nun erste Erfahrungen mit Windows PowerShell gesammelt haben, können Sie die logische Struktur von SharePoint mit Windows PowerShell untersuchen und verschiedene Aspekte einer SharePoint-Implementierung überprüfen. Anschließend sollten Sie in der Lage sein, beispielsweise eine Liste der Blogs aus Ihrer SharePoint-Farm aufzustellen, sortiert nach dem Datum der letzten Änderung. Dazu müssen Sie folgende Arbeitsschritte beherrschen:

- Verwenden der `Get-SP*`-Cmdlets, von denen Objekte zurückgeben werden, die Komponenten aus der logischen Struktur von SharePoint darstellen

- Verwenden der Pipeline zur Erstellung einer Auflistung von Objekten, die Websites der Farm darstellen
- Verwenden des Cmdlets `Select-Object` zur Bearbeitung bestimmter Eigenschaften
- Verwenden des Cmdlets `Where-Object` zur Filterung von Objekten
- Verwenden der `Format-*-` und `Export-*`-Cmdlets zur Erstellung von Berichten

Untersuchen der logischen SharePoint-Struktur mit *Get-SP**

Die logische Struktur einer SharePoint-Farm sieht die Farm, Webanwendungen, Websitesammlungen, Websites, Bibliotheken und Listen vor, von denen sich Letztere wiederum in Ordner, Dokumente und Listenelemente unterteilen lassen. In diesem Kapitel verwenden Sie Cmdlets mit dem Verb `Get`, um Objekte aus den obersten Schichten des SharePoint-Objektmodells abzurufen. Abbildung 2.3 stellt die logische Struktur den entsprechenden `Get-SP*`-Cmdlets gegenüber.

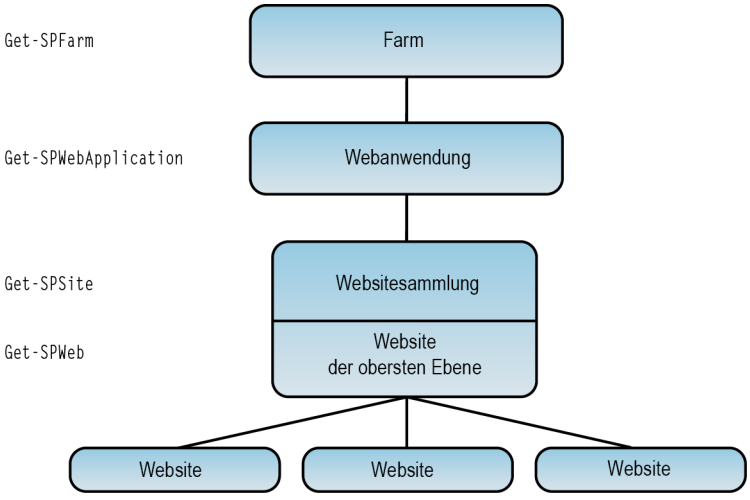

Abbildung 2.3 Die logische Struktur von SharePoint und `Get-SP*`-Cmdlets

Zum Abruf eines Objekts, das die gesamte Farm darstellt, verwenden Sie folgenden Befehl:

```
Get-SPFarm
```

Die Ausgabe dieses Befehls ist der Name der Farm. Standardmäßig handelt es sich dabei um den Namen der SharePoint-Konfigurationsdatenbank.

Um eine Objektauflistung abzurufen, die die Webanwendungen in der Farm darstellt, verwenden Sie folgenden Befehl:

```
Get-SPWebApplication
```

Standardmäßig schließt das Cmdlet `Get-SPWebApplication` die Zentraladministration aus, als Schutz vor Skripts, die jede Webanwendung aus der Farm bearbeiten sollen. Wenn die Zentraladministration ebenfalls berücksichtigt werden soll, geben Sie den Parameter `-IncludeCentralAdministration` an, wie in folgendem Beispiel:

```
Get-SPWebApplication -IncludeCentralAdministration
```

Zur Auflistung aller Websitesammlungen der Farm verwenden Sie folgenden Befehl:

```
Get-SPSite
```

Um den Speicherbedarf und die Bearbeitungszeit überschaubar zu halten, beschränkt das Cmdlet `Get-SPSite` die Zahl der Websitesammlungen, die es zurückgibt, standardmäßig auf 20. Fügen Sie den Parameter `-Limit` hinzu, wenn Sie diese Beschränkung ändern möchten, oder `-Limit All`, um sie aufzuheben. Das Cmdlet `Get-SPSite` berücksichtigt nicht die Zentraladministration, solange die Webanwendung der Zentraladministration nicht als `-Identity`-Parameter angegeben oder in der Pipeline an das Cmdlet übergeben wird.

Anders als `Get-SPSite` oder `Get-SPWebApplication` ruft das Cmdlet `Get-SPWeb` keine Auflistung von Websiteobjekten ab. Ohne Parameter, der einen Bereich festlegt, gibt das Cmdlet `Get-SPWeb` einfach einen Fehler zurück.

`Get-SPWeb` gefolgt von einer URL gibt ein Objekt zurück, das eine einzelne Website repräsentiert, wie in folgendem Bespiel:

```
Get-SPWeb http://intranet.contoso.com/websites/Sales/Blogs
```

`Get-SPWeb` gefolgt vom Parameter `-Site` ruft eine Auflistung der Websites in der angegebenen Websitesammlung ab. Der folgende Befehl ruft beispielsweise die Websites aus der Websitesammlung *Intranet* ab:

```
Get-SPWeb -Site http://intranet.contoso.com
```

Standardmäßig beschränkt das Cmdlet `Get-SPWeb` die Zahl der zurückgegebenen Objekte auf 200. Wie beim Cmdlet `Get-SPSite` können Sie diese Beschränkung mit dem Parameter `-Limit` ändern oder `-Limit All` verwenden, wenn alle Websites aus einer Websitesammlung zurückgegeben werden sollen.

Benutzeroberflächenterminologie versus Objektmodellterminologie

Wie Ihnen sicher bewusst geworden ist, unterscheidet sich die in der Benutzeroberfläche verwendete Terminologie zur Beschreibung der logischen SharePoint-Hierarchie von der Terminologie, die in Windows PowerShell verwendet wird. Die im Objektmodell verwendete Terminologie wird hauptsächlich von Entwicklern und im .NET Framework verwendet und kann auf ein Erbe zurückblicken, das bis in die Anfangszeit von SharePoint zurückreicht.

Insbesondere die Verwendung des Worts *Site* kann zu Missverständnissen führen. Beachten Sie die unterschiedliche Verwendung des Worts Site zur Beschreibung von SharePoint-Komponenten in der Benutzeroberfläche und im Objektmodell.

Benutzeroberfläche und Dokumentation	Objektmodell
Farm	`SPFarm`
Webanwendung	`SPWebApplication`
Websitesammlung	`SPSite`
Website, untergeordnete Website oder Unterwebsite (engl.: site, web site, web, subweb, subsite)	`SPWeb`

Noch schwieriger wird es, wenn ein Benutzer eine Beschwerde wie »Ich kann nicht auf meine Site zugreifen« äußert. Geht es um eine Websitesammlung (`SPSite`), eine Website (`SPWeb`), oder hat der Benutzer *http://intranet.contoso.com* eingegeben und eine Fehlermel-

dung erhalten? Dann könnte das Problem auch in der Webanwendung (`SPWebApplication`) oder im DNS-System liegen.

Es empfiehlt sich, bei Diskussionen über SharePoint, insbesondere bei der Erfassung von Informationen zur Problembehandlung, das Wort *Site* völlig zu vermeiden. Drücken Sie es präziser aus: *Webanwendung* (web application), *Websitesammlung* (site collection) oder *Website* (website).

Schnelltest

- Welche SharePoint-Komponente stellt ein `SPSite`-Objekt in Windows PowerShell dar?

Antwort zum Schnelltest

- Eine Websitesammlung.

Verwenden der Pipeline zur Auflistung eines bestimmten Teils der logischen Struktur

Wie bereits beschrieben, verwendet das Cmdlet `Get-SPWeb` einen `-Site`-Parameter zur Bestimmung der Websitesammlung, aus der Websites zurückgegeben werden sollen. Der folgende Befehl ruft beispielsweise alle Websites aus der Websitesammlung *Intranet* ab.

```
Get-SPWeb -Site "http://intranet.contoso.com" -Limit All
```

Das ebenfalls bereits besprochene Cmdlet `Get-SPSite` ruft alle Websitesammlungen aus der Farm ab. Wenn Sie einen `-Identity`-Parameter verwenden, ruft es nur passende Websitesammlungen ab. Der `-Identity`-Parameter ist der Standardparameter, daher kann der Parametername weggelassen werden. Der folgende Befehl ruft zum Beispiel nur eine Websitesammlung ab:

```
Get-SPSite "http://intranet.contoso.com"
```

Sie können die von `Get-SPSite` zurückgegebene Websitesammlung verwenden, statt `Get-SPWeb` mit dem Parameter `-Site` aufzurufen, sofern Sie die von `Get-SPSite` zurückgegebene Websitesammlung an `Get-SPWeb` weitergeben. Der folgende Befehl ruft zum Beispiel ebenfalls alle Websites aus der Websitesammlung *Intranet* ab:

```
Get-SPSite "http://intranet.contoso.com"| Get-SPWeb -Limit All
```

Sie können die Pipeline verwenden, um ein Objekt, das eine bestimmte Komponente aus der logischen Struktur Ihrer SharePoint-Farm repräsentiert, an ein Cmdlet weiterzugeben, das die im übergebenen Objekt enthaltenen Objekte *aufzählt* (enumerate) oder *auflistet* (list). Um beispielsweise alle Websitesammlungen aus der Webanwendung *Intranet* aufzulisten, geben Sie folgenden Befehl ein:

```
Get-SPWebApplication "http://intranet.contoso.com" | Get-SPSite -Limit All
```

Das Objekt, das von `Get-SPWebApplication` abgerufen wird und die Intranet-Webanwendung darstellt, wird an `Get-SPSite` weitergegeben, das dieses Objekt als `-Identity`-Parameter verwendet.

Durch die Kombination der beiden obigen Befehle entsteht folgender Befehl:

```
Get-SPWebApplication "http://intranet.contoso.com" | Get-SPSite -Limit All |
    Get-SPWeb -Limit All
```

Dieser Befehl listet alle Websites aus der Intranet-Webanwendung auf. Zuerst ruft das Cmdlet `Get-SPWebApplication` ein Objekt ab, das die Intranet-Webanwendung darstellt. Dieses Objekt wird an `Get-SPSite` weitergegeben, das alle in der Intranet-Webanwendung enthaltenen Websitesammlungen abruft. Diese Objekte werden an `Get-SPWeb` weitergegeben, das alle Websites aus den Websitesammlungen auflistet.

Um alle Websites aus der Farm aufzulisten, einschließlich der Zentraladministration, verwenden Sie folgenden Befehl:

```
Get-SPWebApplication -IncludeCentralAdministration | Get-SPSite -Limit All |
    Get-SPWeb -Limit All
```

Ermitteln von Objektmembern (Methoden und Eigenschaften) mit *Get-Member*

Nun können Sie zwar alle Websites einer Farm auflisten, aber es ist an der Zeit, die Websites genauer zu untersuchen. Ziel dieser Beschreibung ist die Identifizierung aller *Blog*-Websites, die es in der Farm gibt, und die Dokumentierung des Datums der letzten Änderung. Dazu brauchen wir ausführlichere Informationen über die einzelnen Websites.

Wie Sie bereits wissen, bearbeiten Windows PowerShell-Cmdlets *Objekte* und geben ein oder mehrere Objekte zurück – Darstellungen der betreffenden Komponenten. Der folgende Befehl liefert beispielsweise ein Objekt, das die Unterwebsite *Blog* der Website *Sales* darstellt:

```
Get-SPWeb "http://intranet.contoso.com/websites/Sales/Blogs"
```

Objekte haben *Member*, wobei *Methoden* (methods) und *Eigenschaften* (properties) die wichtigsten sind. Methoden sind Aktionen, Vorgänge, die Sie mit dem Objekt oder auf das Objekt anwenden können. Eigenschaften sind Attribute. Eine spezielle Art von Eigenschaft ist die *Auflistung* (collection). Sie kann leer sein, ein Element oder mehrere Elemente enthalten.

Wenn Sie herausfinden möchten, welche Methoden und Eigenschaften ein bestimmtes Objekt anbietet, geben Sie das Objekt an das Cmdlet `Get-Member` weiter. Es zeigt die Member eines Objekts an. `Get-Member` erwartet ein Objekt als Eingabe. Die folgenden Befehle listen die Member eines Objekts auf:

```
<Objekt> | Get-Member -MemberType Methods
<Objekt> | Get-Member -MemberType Properties
```

Um beispielsweise die Methoden und Eigenschaften der *Sales Blog*-Website zu sehen, geben Sie folgenden Befehl ein:

```
Get-SPWeb "http://intranet.contoso.com/websites/Sales/Blogs" | Get-Member
```

Das Cmdlet gibt alle Member zurück, einschließlich Methoden, Eigenschaften, Ereignisse und Aliasse. Uns interessieren an dieser Stelle aber nur die Eigenschaften der Website. Sie können die Ausgabe des Cmdlets `Get-Member` eingrenzen, indem Sie den Parameter `-MemberType` mit dem Wert `Properties` für Eigenschaften oder `Methods` für Methoden hinzufügen. Die folgenden Befehle listen die Methoden und Eigenschaften eines Objekts auf:

```
<Objekt> | Get-Member -MemberType Methods
<Objekt> | Get-Member -MemberType Properties
```

Unter den Dutzenden von Eigenschaften eines `SPWeb`-Objekts sind auch `Url` (die Website-Adresse), `WebTemplate` (der Name der Vorlage, mit der die Website erstellt wurde), und `LastItemModifiedDate` (das Datum, an dem die letzte Änderung durchgeführt wurde).

Weitere Informationen *Get-Member*

Der Artikel »Anzeigen der Objektstruktur (Get-Member)« unter *http://technet.microsoft.com/de-de/library/dd315243.aspx* bietet weitere Informationen über das Cmdlet `Get-Member`.

Write-Output

Wenn Sie den Befehl

```
Get-SPWeb "http://intranet.contoso.com/websites/Sales/Blogs"
```

eingeben, wird die URL der Website zurückgegeben. Wie Sie wissen, arbeitet Windows PowerShell zwar mit Objekten, aber wenn ein Cmdlet seine Arbeit beendet hat (am Ende der Pipeline), zeigt ein implizit aufgerufenes `Write-Output`-Cmdlet die Standardeigenschaften des Objekts an. Im gezeigten Beispiel ist die Standardeigenschaft `Url` und das Standardanzeigeformat ist eine Tabelle.

Anzeigen von Eigenschaften mit *Select-Object* (Alias: *Select*)

Sie können mit dem Cmdlet `Select-Object` festlegen, welche Eigenschaften am Ende der Pipeline angezeigt werden sollen. Der Alias dieser Eigenschaft ist `Select`. Um alle Eigenschaften eines Objekts aus der Pipeline anzuzeigen, fügen Sie zum Ende des Befehls `Select *` hinzu. Der folgende Befehl zeigt beispielsweise alle Eigenschaften der Website *Sales Blogs* an:

```
Get-SPWeb "http://intranet.contoso.com/websites/Sales/Blogs" | Select *
```

Welche Eigenschaften angezeigt werden, können Sie festlegen, indem Sie die Namen der gewünschten Eigenschaften nach dem `Select`-Cmdlet angeben. Der folgende Befehl zeigt beispielsweise die URL und die Vorlage der Website *Sales blogs* an:

```
Get-SPWeb "http://intranet.contoso.com/websites/Sales/Blogs" |
    Select URL,WebTemplate,LastItemModifiedDate
```

Wenn Sie den Befehl

```
Get-SPWebApplication | Get-SPSite -limit all | Get-SPWeb -limit all
```

eingeben, werden alle Websites aus der Farm angezeigt, mit Ausnahme der Zentraladministration. Um die URLs und die Vorlagen aller Websites aus der Farm anzuzeigen, fügen Sie das `Select`-Cmdlet am Ende des Befehls an. Geben Sie folgenden Befehl ein:

```
Get-SPWebApplication | Get-SPSite -limit all | Get-SPWeb -limit all |
    Select-Object URL,WebTemplate
```

Praxistipp *Get-SPWebApplication* ist nicht wirklich erforderlich

Dan Holme

Im obigen Befehl ist das Cmdlet `Get-SPWebApplication`, das alle Webanwendungen der Farm mit Ausnahme der Zentraladministration zurückgibt, nicht wirklich erforderlich. Das Cmdlet `Get-SPSite` gibt alle Websitesammlungen der Farm zurück, mit Ausnahme der Zentraladministration, und Websites sind in Websitesammlungen enthalten. Mit folgendem Befehl würde man also dasselbe Ergebnis erhalten:

```
Get-SPSite -limit all | Get-SPWeb -limit all | Select-Object URL,WebTemplate
```

Allerdings wird das Cmdlet `Get-SPWebApplication` aus zwei Gründen in dieser Lektion verwendet. Erstens betont es die logische Struktur von SharePoint und den dazugehörigen `Get-SP*`-Befehlen. Zweitens ist `Get-SPWebApplication` erforderlich, wenn die Zentraladministration berücksichtigt werden soll. Verwenden Sie `Get-SPWebApplication` mit dem Parameter `-IncludeCentralAdministration`, dann wird die resultierende Websiteliste auch die Zentraladministration enthalten.

Weitere Informationen *Select-Object*

Der Artikel »Auswählen von Objektteilen (Select-Object)« unter *http://go.microsoft.com/fwlink/?LinkID=192739* bietet weitere Informationen über das Cmdlet `Select-Object`.

Sortieren der Pipeline mit *Sort-Object* (Alias: *Sort*)

Bei der Arbeit mit umfangreicheren Objektlisten ist es häufig sinnvoll, die Ergebnisse zu sortieren. Zum Sortieren von Objekten können Sie das Cmdlet `Sort-Object` verwenden, dessen Alias `Sort` lautet. Der folgende Befehl zeigt zum Beispiel die URLs und die Vorlagen aller Websites der Farm an, sortiert nach dem Vorlagennamen:

```
Get-SPWebApplication | Get-SPSite -limit all | Get-SPWeb -limit all |
    Select-Object URL,WebTemplate | Sort WebTemplate
```

Sie können den Parameter `-Descending` zum `Sort`-Cmdlet hinzufügen, um in absteigender Reihenfolge zu sortieren. Standardmäßig erfolgt die Sortierung in aufsteigender Reihenfolge und es gibt keinen `-Ascending`-Parameter.

Weitere Informationen *Sort-Object*

Der Artikel »Sortieren von Objekten« unter *http://go.microsoft.com/fwlink/?LinkID=192740* bietet weitere Informationen über das Cmdlet `Sort-Object`.

Anzeigen von Ergebnissen mit *Format-Table* und *Format-List* (Aliasse *ft* und *fl*)

Das Ausgabeformat der Cmdlets hängt von der Zahl der Objekte und der anzuzeigenden Eigenschaften ab. Einige der bisher besprochenen Cmdlets geben Eigenschaften als Listen zurück, andere als Tabellen.

Mit den Cmdlets `Format-List` (alias `fl`) und `Format-Table` (alias `ft`) können Sie das Anzeigeformat ändern. Der folgende Befehl zeigt beispielsweise die URLs und Vorlagen aller Websites der Farm an, sortiert nach dem Vorlagennamen und als Liste formatiert:

```
Get-SPWebApplication | Get-SPSite -limit all | Get-SPWeb -limit all |
    Select-Object URL,WebTemplate | Sort WebTemplate | Format-List
```

Hinweis *Format-List* fügt ein implizites *Select ** hinzu

Durch die Verwendung von `Format-List` (oder `fl`) am Ende der Pipeline wird ein implizites `Select *` hinzugefügt. Alle Eigenschaften werden zurückgegeben. Wenn Sie die Zahl der zurückgegebenen Eigenschaften beschränken möchten, geben Sie die Eigenschaften mit dem `Select`-Cmdlet an.

Weitere Informationen *Format*-Befehle

In dem Artikel »Verwenden von Formatbefehlen für das Ändern der Ausgabeanzeige« unter *http://go.microsoft.com/fwlink/?LinkID=192741* finden Sie weitere Informationen über die `Format-*`-Cmdlets.

Verwalten von Ausgabeformaten

Windows PowerShell kann Objekte speichern, exportieren und in eine ganze Reihe von Formaten konvertieren. Zu den nützlichsten zählen:

- CSV-Dateien (Comma-Separated Value)
- XML-Dateien (Extensible Markup Language)
- Die Rasteransicht (GridView)

Um Ausgaben in einer CSV-Datei zu speichern, fügen Sie `| Export-CSV <Dateiname>` zum Ende der Pipeline hinzu.

Sie können `| ConvertTo-XML` zum Ende der Pipeline hinzufügen, um die Ausgabe in ein XML-Objekt zu konvertieren. Ein XML-Objekt lässt sich aber nicht direkt anzeigen, weil es ein Objekt ist, nicht die Textausgabe einer XML-Datei. Daher müssen Sie die Pipeline und damit auch das XML-Objekt speichern. Folgen Sie diesem Beispiel:

```
( <Befehl> | ConvertTo-XML ).Save("<Dateiname>")
```

Der folgende Befehl erstellt zum Beispiel eine XML-Datei, in der die URLs und Vorlagen aller Websites der Farm nach Vorlagennamen sortiert aufgeführt werden:

```
(Get-SPWebApplication | Get-SPSite -limit all | Get-SPWeb -limit all |
    Select-Object URL,WebTemplate | Sort WebTemplate |
    ConvertTo-XML).Save("C:\Users\SP_Admin\Desktop\SharePointWebsiteTemplates.xml")
```

Zu Windows PowerShell 2.0 gehört eine integrierte Skriptumgebung, die eine interaktive Anzeige von Tabellen ermöglicht. Sorgen Sie dafür, dass das Feature *Windows PowerShell Integrated Scripting Environment (ISE)* installiert ist. Sollte dies nicht der Fall sein, öffnen Sie den Server-Manager und fügen es mit dem *Assistenten zum Hinzufügen von Features* hinzu.

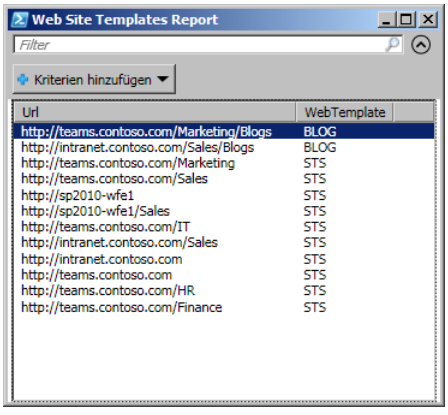

Abbildung 2.4 Anzeige von Daten als Tabelle (DataGrid)

Das folgende Beispiel zeigt die Daten als Tabelle an:

```
Get-SPWebApplication | Get-SPSite -limit all | Get-SPWeb -limit all |
    Select-Object URL,WebTemplate | Sort WebTemplate |
    Out-GridView -Title "Web Site Templates Report"
```

Abbildung 2.4 zeigt die resultierende Tabelle.

Weitere Informationen *Out- Cmdlets**

Der Artikel »Umleiten von Daten mit `Out-*`-Cmdlets« unter *http://go.microsoft.com/fwlink/?LinkID=192742* bietet weitere Informationen über die `Out-*`-Cmdlets.

Filtern der Pipeline mit *Where-Object* (Aliasse: *Where*, *?*)

Manchmal ist eine Beschränkung auf eine Teilmenge der abgerufenen Objekte erforderlich. Im vorigen Abschnitt hat das Cmdlet `Get-SPWeb` zum Beispiel alle Websites zurückgegeben. Sie haben das Ergebnis zwar nach den Namen der Vorlagen sortiert, mit denen die Websites erstellt wurden, aber was ist zu tun, wenn Sie nur die Websites brauchen, die mit der *Blog*-Websitedefinition erstellt wurden?

Das Cmdlet `Where-Object` filtert die in der Pipeline vorhandenen Objekte. Die nachfolgenden Cmdlets in der Pipeline haben nur noch die Objekte zur Verfügung, die es durch den Filter geschafft haben. Der folgende Befehl ruft beispielsweise alle Websites der Farm ab, die mit der *Blog*-Websitedefinition erstellt wurden. Dabei wird die Eigenschaft `WebTemplate` des `SPWeb`-Objekts ausgewertet:

```
Get-SPWebApplication | Get-SPSite -Limit All | Get-SPWeb -Limit All |
    Where-Object { $_.WebTemplate -eq "BLOG"}
```

Der Filter – das zur Überprüfung der Websites verwendete Kriterium – ist ein *Ausdruck*, der von geschweiften Klammern eingeschlossen wird. Der Ausdruck besagt, dass die `WebTemplate`-Eigenschaft eines Objekts den Wert »BLOG« enthalten muss. Erfüllt eine Website dieses Kriterium nicht, wird das Objekt aus der Pipeline entfernt.

Beachten Sie die Verwendung der speziellen Variablen `$_`. Sie steht für das aktuelle Objekt in der Pipeline. Von `Get-SPWeb` wird eine Auflistung mit allen Websites der Farm an `Where-Object` weitergegeben. Die Aufgabe von `Where-Object` ist es, die Pipeline zu filtern und nur die Websites weiterzugeben, die mit der *Blog*-Vorlage erstellt wurden. Man kann sich leicht vorstellen, dass das `Where-Object` alle Websiteobjekte in einer Schleife untersucht und dabei überprüft, ob sie das Kriterium erfüllen. Die spezielle Variable `$_` repräsentiert in jedem Schleifendurchlauf von `Where-Object` die jeweils aktuelle Website.

Weitere Informationen Filterung mit *Where-Object*

Im Artikel »Entfernen von Objekten aus der Pipeline (Where-Object)« unter *http://go.microsoft.com/fwlink/?LinkID=192743* finden Sie weitere Informationen über das Cmdlet `Where-Object`.

Serverseitige Filterung mit dem Parameter *-Filter*

Wenn Sie die Pipeline mit dem Cmdlet `Where-Object` filtern, ruft Windows PowerShell alle Objekte vom Server ab und filtert die Objekte dann lokal auf dem Clientcomputer. Im zuvor gezeigten Beispiel werden alle Objekte vom Cmdlet `Get-SPWeb` abgerufen, und dann muss der Windows PowerShell-Client die Objekte filtern. Sie können die Belastung des Servers verringern, indem Sie eine serverseitige Filterung verwenden, wann immer es möglich ist. Einige Cmdlet unterstützen einen `-Filter`-Parameter für eine serverseitige Filterung.

Das `SPWeb`-Objekt kann serverseitig nach den Eigenschaften `Title` und `Template` gefiltert werden. Die Objekte `SPSite` und `SPSiteAdministration` lassen sich serverseitig nach `Owner`, `SecondaryContact` und `LockState` filtern.

Da Sie in diesem Beispiel die Möglichkeit haben, eine serverseitige Filterung zu verwenden, sollten Sie dies auch tun. Der folgende Befehl ruft zum Beispiel alle Websites ab, die mit der *Blog*-Websitedefinition erstellt wurden, und verwendet dabei eine serverseitige Filterung der `SPWeb`-Objekte:

```
Get-SPWebApplication | Get-SPSite -Limit All | Get-SPWeb -Limit All
    -Filter {$_.Template -eq "BLOG#0"}
```

Vielleicht ist Ihnen aufgefallen, dass der Ausdruck im serverseitigen Filter überprüft, ob die Eigenschaft `Template` den Wert »BLOG#0« enthält. Darin unterscheidet sich der Filter von dem Filter, den wir auf dem Client verwendet haben, denn darin wurde überprüft, ob die Eigenschaft `WebTemplate` den Wert »BLOG« aufweist. »BLOG« ist die `Title`-Eigenschaft der Vorlage, »BLOG#0« ist ihr `Name`.

Operatoren

In den bisher verwendeten Filterausdrücken ist Ihnen vielleicht der Vergleichsoperator `-eq` aufgefallen. Er steht für *equals* oder »ist gleich«. Die folgenden Operatoren lassen sich in Ausdrücken verwenden:

- ***-lt*** kleiner als
- ***-le*** kleiner als oder gleich
- ***-gt*** größer als
- ***-ge*** größer als oder gleich
- ***-eq*** ist gleich
- ***-ne*** ist nicht gleich
- ***-like*** wie; kann Platzhalter für einen Mustervergleich verwenden
- ***-and*** logisches Und
- ***-or*** logisches Oder

Aufbau einer effektiven Pipeline

Für die Arbeit mit Objekten, die durch die Pipeline von Windows PowerShell weitergereicht werden, hat sich eine bestimmte Reihenfolge der Arbeitsschritte als sinnvoll erwiesen:

- ***Abrufen*** Verwenden Sie das Verb `Get`, um Objekte abzurufen.
- ***Filtern*** Verwenden Sie das `Where`-Cmdlet, um die Objekte herauszufiltern, mit denen Sie in der Pipeline arbeiten möchten. In den wenigen Fällen, in denen eine serverseitige Filterung möglich ist, fügen Sie zu den `Get`-Befehlen einen `-Filter`-Parameter hinzu.

- ***Bearbeiten*** Sie tun etwas mit den Objekten, wobei Sie die Cmdlets verwenden, die sich für die in der Pipeline verbliebenen Objekte eignen.
- ***Auswählen*** Wählen Sie mit dem `Select`-Cmdlet die Eigenschaften der Objekte aus, die ausgegeben oder angezeigt werden sollen.
- ***Sortieren*** Sortieren Sie die Ergebnisse vor der Ausgabe mit dem `Sort`-Cmdlet.
- ***Ausgeben*** Bringen Sie die Ausgabe mit den `Format-*`-, `Export-*`- und `Out-*`-Cmdlets ins gewünschte Format. Wenn Sie die in der Pipeline verbliebenen Objekte in ein bestimmtes Format konvertieren möchten, können Sie dies mit den `Convert-*`-Cmdlets tun. Speichern Sie die Objekte anschließend mit der `Save`-Methode der Pipeline in einer Datei. In einem der bereits gezeigten Beispiele wird die Ausgabe in ein XML-Objekt konvertiert und dann in einer XML-Datei gespeichert.

Diese Liste führt die Pipelineaktivitäten zwar in einer typischen und effektiven Reihenfolge auf, aber in vielen Windows PowerShell-Befehlen sind gar nicht alle genannten Aktivitäten erforderlich.

Sehen Sie sich das folgende Beispiel an:

```
Get-SPWebApplication | Get-SPSite -Limit ALL | Get-SPWeb -Limit ALL |
    Where-Object { $_.WebTemplate -eq "BLOG"} |
    Select URL,Title,WebTemplate,LastItemModifiedDate | Sort LastItemModifiedDate |
    Export-CSV desktop\BlogActivityReport.csv
```

Dieser Befehl macht Folgendes:

- Er ruft alle Objekte ab, nämlich alle Websites einer Farm.

  ```
  Get-SPWebApplication | Get-SPSite -Limit ALL | Get-SPWeb -Limit ALL
  ```

- Er filtert die Pipeline, damit nur Websites übrig bleiben, die mit der *Blog*-Vorlage erstellt wurden.

  ```
  Where-Object { $_.WebTemplate -eq "BLOG"}
  ```

- Er wählt bestimmte Eigenschaften der Websites aus.

  ```
  Select URL,Title,WebTemplate,LastItemModifiedDate
  ```

- Er sortiert das Ergebnis nach dem Datum der letzten Änderung eines Elements in der Website.

  ```
  Sort LastItemModifiedDate
  ```

- Er exportiert die Ergebnisse in eine *.csv*-Datei.

  ```
  Export-CSV desktop\BlogActivityReport.csv
  ```

Erstellen eines SharePoint-Intranets mit Windows PowerShell

Sie können die SharePoint-Komponenten, die Sie in Kapitel 1 mit der Zentraladministration erstellt haben, auch mit Windows PowerShell erstellen.

Erstellen einer Webanwendung mit *New-SPWebApplication*

Das folgende Beispiel zeigt, wie man mit dem Cmdlet `New-SPWebApplication` eine neue Webanwendung erstellen kann:

```
New-SPWebApplication -Name <Name> -Port <Port> -HostHeader <Hostheader> -URL <URL>
    -ApplicationPool <Anwendungspool> -ApplicationPoolAccount <Anwendungspoolkonto>
    -DatabaseName <Datenbankname>
```

Darin gilt:

- *<Name>* ist der Name der neuen Webanwendung.
- *<Port>* ist der Port, an den die Webanwendung in IIS gebunden wird.
- *<Hostheader>* ist der Hostheader im Format *server.domäne.com*.

  Beachten Sie, dass die `Get-Help`-Dokumentation des Cmdlets für den Hostheader das Format *http://server.domäne.com* angibt. Das ist nicht korrekt.
- *<URL>* ist die öffentliche URL (mit Lastausgleich) für die Webanwendung.
- *<Anwendungspool>* ist der Name des Anwendungspools.
- *<Anwendungspoolkonto>* ist das verwaltete Konto, das der Anwendungspool verwenden soll.

  Dieser Parameter ist erforderlich, wenn Sie einen Anwendungspool angeben, der noch nicht vorhanden ist. Verwenden Sie das Cmdlet `Get-SPManagedAccount`, wie im folgenden Beispiel. Sofern der Anwendungspool bereits vorhanden ist, verwenden Sie diesen Parameter nicht.
- *<Datenbankname>* ist der Name der ersten Inhaltsdatenbank für die Webanwendung.

Der folgende Befehl erstellt beispielsweise eine Intranet-Webanwendung mit einer Konfiguration, die dem Intranet ähnelt, das in Kapitel 1 mit der Zentraladministration erstellt wurde.

```
New-SPWebApplication -Name "Contoso Intranet" -Port 80
    -HostHeader "intranet.contoso.com" -URL "http://intranet.contoso.com:80"
    -ApplicationPool "SharePoint Web Applications"
    -ApplicationPoolAccount (Get-SPManagedAccount "CONTOSO\SP_WebApps")
    -DatabaseName "SharePoint_Content_Intranet"
```

Der Befehl erstellt einen neuen Anwendungspool. Falls der gewünschte Anwendungspool bereits vorhanden ist, lassen Sie den Parameter `-ApplicationPoolAccount` und seinen Wert weg.

Löschen einer Webanwendung

Zum Löschen einer Webanwendung verwenden Sie das Cmdlet `Remove-SPWebApplication`. Die folgende Zeile löscht zum Beispiel die Intranet-Webanwendung, einschließlich der IIS-Website und der Inhaltsdatenbanken:

```
Remove-SPWebApplication http://intranet.contoso.com
    -DeleteIISSite -RemoveContentDatabase -Confirm:$false
```

Das Verb zum Löschen einer Webanwendung ist `Remove`, nicht `Delete`. `Delete` ist in Windows PowerShell gar kein gültiges Verb. Für jedes Objekt, das Sie in dieser Lektion mit einem `Create-SP*`-Cmdlet erstellen, gibt es ein entsprechendes `Remove-SP*`-Cmdlet.

Beachten Sie den Parameter `-Confirm:$false`. Der Parameter `-Confirm` ist in allen Windows PowerShell-Befehlen üblich, die potenziell schädliche Wirkungen haben. Die Standardeinstellung ist, einen Vorgang zu bestätigen (`-Confirm:$true`). Das Cmdlet fordert den Benutzer dann zur Bestätigung des Vorgangs auf. Mit `-Confirm:$false` können Sie solche Bestätigungsaufforderungen unterdrücken.

Außerdem können Sie den Parameter `-WhatIf` für einen Probelauf verwenden, um die Ausführung des Befehls zu simulieren und eine Meldung über dessen Wirkung anzuzeigen. Besonders nützlich ist der `-WhatIf`-Parameter bei der Arbeit mit einer unbekannten Zahl von Objekten oder mit einer Auflistung von Objekten, damit Sie genau sehen, was mit welchen Objekten geschieht.

Erstellen einer Websitesammlung mit *New-SPSite*

Das folgende Beispiel zeigt die Verwendung des Cmdlets `New-SPSite` zur Erstellung einer neuen Websitesammlung.

```
New-SPSite -Url "<URL für die neue Websitesammlung>"
    -ContentDatabase <Name der Inhaltsdatenbank>
    -Name "<Name für die Website der obersten Ebene>"
    -OwnerAlias "<Domäne\Benutzer>" -Template <Vorlage>
```

Darin gilt:

- *<URL>* ist die URL der zu erstellenden Websitesammlung.
- *<Name der Inhaltsdatenbank>* ist der Name der Inhaltsdatenbank, in der die Websitesammlung erstellt werden soll. Dieser Parameter ist optional.
- *<Name>* ist der Name der Website in der obersten Ebene. Der Name wird im Titel und in der Kopfzeile der Website der obersten Ebene angezeigt.
- Der Wert *<Domäne\Benutzer>* des Parameters `-OwnerAlias` legt den primären Websitesammlungsadministrator fest. Der sekundäre Websitesammlungsadministrator wird mit dem Parameter `-SecondaryOwnerAlias` festgelegt.
- *<Vorlage>* ist die Websitedefinition für die Website der obersten Ebene, beispielsweise `BLANKINTERNET#0`, die *Veröffentlichungssite*, oder `STS#0`, die *Teamwebsite*.

Der folgende Befehl erstellt zum Beispiel im Stamm der Intranet-Webanwendung eine Websitesammlung und erstellt dann mit der Websitedefinition *Veröffentlichungssite* eine Website der obersten Ebene.

```
New-SPSite -Url "http://intranet.contoso.com" -Name "Contoso Intranet"
    -OwnerAlias "CONTOSO\SP_Admin" -Template "BLANKINTERNET#0"
```

Auflisten der verfügbaren Websitedefinitionen

Woher wissen Sie, was als Wert für *<Vorlage>* anzugeben ist? Geben Sie den folgenden Befehl ein, um eine Liste der verfügbaren Websitedefinitionen zu erstellen:

```
Get-SPWebTemplate
```

In der Spalte *Name* steht der Wert, den Sie bei der Konfiguration der Stammwebsite einer neuen Websitesammlung oder bei der Erstellung einer neuen Website verwenden.

Erstellen einer Website

Das folgende Beispiel zeigt, wie sich mit dem Cmdlet `New-SPWeb` eine neue Website erstellen lässt.

```
New-SPWeb <Identität> -Name <Name> -Template "<Vorlage>"
```

Darin gilt:

- *<Identität>* ist die URL der neuen Website.
- *<Name>* ist der Name der Website.
- *<Vorlage>* gibt die Websitedefinition für die Website an, beispielsweise `BLANKINTERNET#0`, die *Veröffentlichungssite*, `BLOG#0`, die *Blog*-Website, oder `STS#0`, die *Teamwebsite*.

Der folgende Befehl erstellt auf der Intranet-Website eine neue Unterwebsite für den Firmenblog:

```
New-SPWeb "http://intranet.contoso.com/Blogs" -Name "Contoso Blogs" -Template "BLOG#0"
```

Variablen

Wenn Sie damit beginnen, Windows PowerShell-Befehle und Skripts für die Praxis zusammenzustellen und zu entwickeln, müssen Sie das Konzept der *Variablen* kennen. Variablen bezeichnen Orte im Speicher, an denen man einen Wert oder ein Objekt ablegen kann. In Windows PowerShell haben Variablen einen Namen, der mit dem Dollarzeichen ($) beginnt.

Um eine Variable zu definieren und ihr einen Wert zuzuweisen, verwenden Sie einfach folgende Syntax:

```
$Variable = Wert
```

Das folgende Skript hält zum Beispiel den Zeitgeberdienst von SharePoint an:

```
$service = "SPTimerV4"
Get-Service $service | Stop-Service
```

Das Ergebnis ist dasselbe wie bei dem bereits gezeigten Einzeiler. Allerdings lässt sich ein Skript später leichter ändern, wenn der Name des Dienstes von der Zeile getrennt wird, die den Dienst abruft und beendet. Sie könnten auch mit dem Cmdlet `Read-Host` den Benutzer nach dem Namen des Prozesses fragen, statt den Namen im Skript vorzugeben.

Bei der Zuweisung einer Zeichenkette an eine Variable schließen Sie den Wert in einfache oder doppelte Anführungszeichen ein, wie bereits gezeigt. Variablen können auch ein Objekt oder mehrere aufnehmen. Untersuchen Sie das folgende Skript:

```
$websites = Get-SPSite http://intranet.contoso.com -Limit ALL | Get-SPWeb -Limit ALL
$websites | Select name, URL, WebTemplate
```

In diesem Beispiel nimmt die Variable `$websites` eine Auflistung der Websites aus der Intranet-Webanwendung auf. In der nächsten Zeile wird die Variable dann verwendet, um die Namen, die URLs und die Vorlagen der Websites anzuzeigen.

In Windows PowerShell gibt es auch einige vordefinierte Variablen wie:

- ***$true*** Boolescher Wert für »wahr«
- ***$false*** Boolescher Wert für »falsch«
- ***$error*** Enthält das Fehlerobjekt, das beim letzten Fehler erstellt wurde

Windows PowerShell verfügt auch über automatische Variablen wie die Variable `$_`, die Sie in dieser Lektion bereits kennengelernt haben.

Weitere Informationen Variablen

Unter *http://go.microsoft.com/fwlink/?LinkID=192734* finden Sie im Artikel »Speichern von Objekten mit Variablen« weitere Informationen über Variablen.

Schleifen mit *ForEach-Object* (Aliasse: *%*, *ForEach*)

Eine der Stärken von Windows PowerShell ist die Leichtigkeit, mit der sich ein Bearbeitungsvorgang auf mehrere Objekte anwenden lässt. Viele Cmdlets durchlaufen automatisch entsprechend viele Schleifen, wenn mehrere Objekte in der Pipeline zu bearbeiten sind (anders gesagt: sie »iterieren«). Sehen Sie sich folgendes Beispiel an:

```
Get-SPWebApplication | Get-SPSite -Limit ALL | Get-SPWeb -Limit ALL
```

Dieser Befehl gibt alle Websites aus der Farm zurück, mit Ausnahme der Zentraladministration. Das Cmdlet `Get-SPWeb` erhält vom Cmdlet `Get-SPSite` eine Auflistung von `SPSite`-Objekten (Websitesammlungen). Es durchläuft die Auflistung der Websitesammlung in einer Schleife und gibt jeweils eine Auflistung der Websites zurück, die sich in der untersuchten Websitesammlung befinden.

Einige Cmdlets können allerdings nicht mehrere Objekte bearbeiten. Das Cmdlet `Enable-SPFeature` zum Beispiel aktiviert ein SharePoint-Feature in einem bestimmten Bereich, etwa in der Website, der Websitesammlung oder der Farm. Der Bereich wird mit dem `-URL`-Parameter des Cmdlets angegeben. Um beispielsweise das Bewertungsfeature für die *Contoso Intranet*-Websitesammlung zu aktivieren, geben Sie folgenden Befehl ein:

```
Enable-SPFeature "Ratings" -Url "http://intranet.contoso.com"
```

Dieses Cmdlet kann nicht mehr als ein Objekt bearbeiten. Wenn Sie also das Bewertungsfeature für alle Websitesammlungen in der Farm aktivieren möchten, können Sie nicht folgenden Befehl verwenden:

```
Get-SPWebApplication | Get-SPSite -Limit ALL | Enable-SPFeature "Ratings"
```

Der Befehl würde die ihm zugedachte Aufgabe nicht erfüllen, weil er die Objekte in der Pipeline – die von `Get-SPSite` zurückgegebenen Websitesammlungen – weder erwartet noch versteht.

Deswegen ist das Cmdlet `ForEach-Object` eines der wichtigsten Cmdlets, wenn es um die Arbeit mit mehreren Objekten geht. Für `ForEach-Object` gibt es einen oft benutzten Alias namens `ForEach` und einen superkurzen Alias `%`.

Das Cmdlet `ForEach-Object` bearbeitet in einer Schleife jedes Objekt in der Pipeline und führt für jedes Objekt einen oder mehrere Vorgänge aus, die in einem von geschweiften Klammern begrenzten *Skriptblock* angegeben werden.

Der folgende Befehl aktiviert zum Beispiel das Bewertungsfeature für alle Websitesammlungen in der Farm:

```
Get-SPWebApplication | Get-SPSite -Limit ALL |
    ForEach-Object { Enable-SPFeature "Ratings" -Url $_.url }
```

Beachten Sie die Verwendung der speziellen Variablen `$_`. Auch hier repräsentiert sie das aktuelle Objekt in der Pipeline. Das Cmdlet `ForEach-Object` durchläuft alle Objekte, die sich in der Pipeline befinden, und übergibt sie der Reihe nach einzeln an das Cmdlet `Enable-SPFeature`. Das Cmdlet `Enable-SPFeature` erwartet eine URL als Bereichsangabe. Daher wird die `URL`-Eigenschaft der aktuellen Websitesammlung im Parameter `-Url` übergeben.

Vielleicht fällt Ihnen eine gewisse Ähnlichkeit mit dem bereits besprochenen Cmdlet `Where-Object` auf. `Where-Object` wendet einen Filter auf alle Objekte an, die sich in der Pipeline befinden, und die Syntax der gezeigten Beispiele ist ähnlich, wobei geschweifte Klammern einen Ausdruck einschließen, der die spezielle Variable `$_` verwendet.

Weitere Informationen **Iteration mit *ForEach-Object***

Unter *http://go.microsoft.com/fwlink/?LinkID=192744* finden Sie im Artikel »Wiederholen einer Aufgabe für mehrere Objekte (ForEach-Object)« weitere Informationen über die Programmierung von Schleifen mit `ForEach-Object`.

Erstellen von mehreren Websites mit einem Windows PowerShell-Skript

Sehen Sie sich das folgende Skript an. Es erstellt die Intranet-Websites HR, Marketing und Finance:

```
$departments = ("HR", "Marketing", "Finance")
ForEach($dept in $departments)
{
    New-SPWeb -Url http://intranet.contoso.com/$dept -Name "$dept" -Template "STS#0"
}
```

Untersuchen Sie dieses Skript nun Zeile für Zeile.

```
$departments = ("HR", "Marketing", "Finance")
```

Diese Zeile erstellt ein *Array*. Das ist ein Datentyp, der mehrere Elemente aufnehmen kann, in diesem Fall Zeichenfolgen. Die Arrayelemente werden durch Kommas voneinander getrennt. Die Klammern und die Elemente sind optional, erleichtern aber das Lesen. Das Array wird einer Variablen namens `$departments` zugewiesen. Wie Sie wissen, beginnt der Name einer Variablen immer mit einem Dollarzeichen ($) und ist frei wählbar, mit Ausnahme der reservierten Wörter (beispielsweise `true`).

```
ForEach($dept in $departments)
```

Mit dieser Zeile beginnen die Schleifen. `ForEach` führt den Skriptblock für jeden Wert aus, der sich in der Arrayvariablen `$departments` befindet. In jedem Schleifendurchlauf wird das aktuelle Element der Variablen `$dept` zugewiesen. In jedem Schleifendurchlauf enthält `$dept` also die aktuelle Zeichenfolge, nämlich den aktuellen Abteilungsnamen.

Der Skriptblock wird in geschweifte Klammern eingeschlossen.

```
{
```

Mit der linken geschweiften Klammer beginnt der Skriptblock.

```
New-SPWeb -Url http://intranet.contoso.com/$dept -Name "$dept" "-Template "STS#0"
```

Die Variable `$dept` wird dazu verwendet, jeder Abteilungswebsite eine eindeutige URL zuzuweisen. Daher ist sie das letzte Element der URL. Außerdem wird die Variable `$dept` dazu verwendet, jeder Website einen eindeutigen Namen zu geben.

```
}
```

Die rechte geschweifte Klammer beendet den Skriptblock.

Am Ende des Skripts steht eine Leerzeile. Wenn Sie das Skript direkt in der Windows PowerShell-Konsole eingeben, müssen Sie zum Schluss eine Leerzeile eingeben, damit die Ausführung des Skripts beginnt.

Schleifen mit der Anweisung *ForEach*

Ein Geständnis: Ich habe ein neues Element eingeführt, ohne Sie darauf hinzuweisen. Im obigen Skript haben wir zwar `ForEach` verwendet, aber das ist *nicht* der `ForEach`-Alias für das Cmdlet `ForEach-Object`, das Sie bereits kennengelernt haben. Es ist die `ForEach`-Anweisung.

Der Unterschied ist klein, aber wichtig. Das Cmdlet `ForEach-Object` oder sein Alias `ForEach` wird gewöhnlich in der Mitte der Pipeline ausgeführt, durchläuft eine Gruppe von Objekten, die sich in der Pipeline befinden, und führt für jedes dieser Objekte den Code im Skriptblock aus. Die `ForEach`-Anweisung steht gewöhnlich am Anfang der Pipeline, stellt eine Objektgruppe zusammen oder verwendet eine vorhandene Objektgruppe und führt für jedes dieser Objekte den Code im Skriptblock aus. `ForEach-Object` erhält die zu bearbeitenden Objekte also über die Pipeline, während die `ForEach`-Anweisung mit ihren eigenen Objekten arbeitet, in diesem Beispiel mit einem Array, das Zeichenfolgen mit den Abteilungsnamen enthält.

Als Einsteiger in Sachen Windows PowerShell brauchen Sie sich vorläufig nur zu merken, dass `ForEach` eine Anweisung oder ein Cmdlet sein kann und dass die Wirkung der beiden Sprachkonstrukte sehr ähnlich ist. Die kleinen Unterschiede zwischen der Anweisung und dem Cmdlet werden deutlich, wenn Sie sich länger mit Windows PowerShell beschäftigen.

Windows PowerShell-Skripts

Windows PowerShell-Skripts können, wie Sie bereits wissen, direkt in der Konsole eingegeben werden. Üblich ist aber, Skripts als Textdateien mit der Dateinamensendung *.ps1* zu speichern.

Lesen und Erstellen von Skripts

Wenn Sie Windows PowerShell-Skripts lesen, die andere geschrieben haben, werden Sie feststellen, dass so manches Skript nicht einfach zu lesen und zu verstehen ist. Manche Leute machen sich einen Spaß daraus, Skripts unübersichtlicher zu strukturieren, als nötig wäre, und erstellen beispielsweise komplizierte Einzeiler, in denen der gesamte Code auf einer einzigen Zeile steht, wobei ein Semikolon (;) die Codeabschnitte an den passenden Stellen voneinander trennt.

In Windows PowerShell dient das Semikolon dazu, separate Befehle zu einer einzigen Zeile zusammenzufassen. Der folgende Befehl (Sie müssen ihn in einer einzigen Zeile eingeben, auch wenn er hier umbrochen ist) erstellt zum Beispiel im Contoso-Intranet Websites für zwei Abteilungen:

```
$departments = ("Manufacturing", "Purchasing") ; ForEach($dept in $departments)
    { New-SPWeb -Url http://intranet.contoso.com/$dept -Name "$dept" -Template "STS#0" }
```

Wie Sie sehen, ist ein Skript durch die Zusammenfassung der Zeilen schwerer zu lesen. Es empfiehlt sich, separate Befehle in separate Zeilen zu schreiben.

Manche Leute verwenden auch zu viele Aliasse, wodurch es für andere schwierig wird, das Skript zu verstehen. Das gilt besonders für Aliasse, die nur aus einem oder aus zwei Buchstaben bestehen, wie `%` (`ForEach-Object`) und `?` (`Where-Object`). Aliasse lassen sich in der Konsole zwar schneller als der ausgeschriebene Cmdlet-Name eingeben, verschlechtern aber die Lesbarkeit. Wenn Sie einen Befehl nur einmal verwenden, reicht ein Alias. Aber wenn Sie einen Befehl dokumentieren oder ein Skript erstellen, sollten Sie den vollständigen Cmdlet-Namen oder einen leicht verständlichen Alias verwenden, damit Sie Ihre eigenen Skripts auch später noch verstehen. Vergessen Sie nicht, dass Sie sich die Eingabe der Cmdlet- und Parameter-

namen in der Konsole erleichtern können, indem Sie die automatische Vervollständigung mit der TABULATORTASTE verwenden.

Ausführen von Windows PowerShell-Skripts

Standardmäßig lässt Windows PowerShell keine Ausführung von Skripts zu. Sinn dieser Maßnahme ist es, eine Beschädigung des Computers oder der Umgebung durch bösartige Skripts zu verhindern. Sie werden diese Standardeinstellung ändern müssen, um Skripts ausführen zu können.

Die Windows PowerShell-Eigenschaft `ExecutionPolicy` entscheidet, welche Skripts ausgeführt werden können. `ExecutionPolicy` lässt sich auf einen der folgenden Werte einstellen:

- **Restricted** Es werden keine Skripts ausgeführt. Das ist die Standardeinstellung.
- **AllSigned** Nur signierte Skripts werden ausgeführt. Sämtliche Skripts müssen von einem vertrauenswürdigen Herausgeber signiert werden, auch die Skripts, die Sie selbst schreiben.
- **RemoteSigned** Aus dem Internet heruntergeladene Skripts müssen von einem vertrauenswürdigen Herausgeber signiert worden sein. Andere Skripts können ohne Signatur ausgeführt werden. Bei der Installation von SharePoint 2010 wird die Standardeinstellung der Ausführungsrichtlinie auf `RemoteSigned` geändert.
- **Unrestricted** Alle Skripts können ausgeführt werden. Allerdings werden Sie zu einer Bestätigung aufgefordert, wenn Sie ein nicht signiertes Skript ausführen möchten, das aus dem Internet heruntergeladen wurde.
- **Bypass** Alle Skripts können ausgeführt werden. Es findet keine Blockierung statt, es werden keine Warnungen ausgegeben und Sie werden nicht zu einer Bestätigung aufgefordert.
- **Undefined** Die Ausführungsrichtlinie wird aus dem aktuellen Bereich entfernt. Dadurch wird aber keine Ausführungsrichtlinie entfernt, die in den Gruppenrichtlinien festgelegt wurde.

Mit folgendem Befehl überprüfen Sie die Einstellungen der Ausführungsrichtlinie:

```
Get-ExecutionPolicy
```

Damit lokal erstellte Skripts ausgeführt werden können, geben Sie folgenden Befehl ein:

```
Set-ExecutionPolicy -RemoteSigned
```

Wenn Sie die Ausführung von Skripts zulassen, die Vertrauensstufe durch den Verzicht auf Signaturen herabsetzen und Warnungen und Blockierungen deaktivieren, steigt natürlich das Sicherheitsrisiko.

Bei den ersten Versuchen, Windows PowerShell-Skripts auszuführen, werden Sie wahrscheinlich auf die üblichen Schwierigkeiten stoßen. Die erste kleine Herausforderung besteht darin, dass der Name eines Skripts nicht reicht, um es auszuführen. Sie müssen auch den Pfad zum Skript kennen.

Sie können ein Skript in einem Ordner speichern, der über die `Path`-Variable zu finden ist. Geben Sie **$env:path** ein und untersuchen Sie die Umgebungsvariable `Path`. Wenn ein Skript über die Angaben in `Path` zu finden ist, reicht der Name des Skripts aus, um es zu starten. Als Alternative bietet es sich an, das Skript im aktuellen Ordner zu speichern und **.\<Skriptdateiname>** einzugeben, oder geben Sie den vollständigen Pfadnamen (Pfad und Dateiname) des Skripts ein.

Wenn der angegebene Pfad oder der Dateiname eines Skripts ein Leerzeichen enthält, müssen Sie den Pfadnamen in Anführungszeichen schreiben und die `Call`-Anweisung voranstellen, oder ihren Alias `&`, wie im folgenden Beispiel:

```
& "C:\Skripts\Mein Ordner\Skript.ps1"
```

Zur Ausführung eines Skripts in einer Eingabeaufforderung, mit einem Run-Befehl oder mit einer geplanten Aufgabe orientieren Sie sich an folgender Vorlage:

```
powershell.exe -noexit &'Pfad\Skript.ps1'
```

Der Parameter `-noexit` bewirkt, dass die Windows PowerShell-Konsole nach der Ausführung des Skripts geöffnet bleibt, damit Sie das Ergebnis des Skripts überprüfen können. Der `-noexit`-Parameter muss direkt auf den Befehl *Powershell.exe* folgen. Wenn Sie die Ausgabe des Skripts gar nicht sehen möchten, lassen Sie den Parameter einfach weg.

Hinweis Es sind weitere Snap-Ins erforderlich

Wenn Sie *Powershell.exe* starten, wird Windows PowerShell mit der Standardkonfigurationsdatei gestartet, die kein Laden des SharePoint-Snap-Ins vorsieht. Im Verlauf dieser Lektion wird noch beschrieben, wie sich das SharePoint-Snap-In laden lässt. In einem Skript, das SharePoint-Cmdlets verwendet, sollten Sie überprüfen, ob das Snap-In geladen ist, und es gegebenenfalls laden, damit sich das Skript auch in einer Windows PowerShell-Konsole mit Standardkonfiguration ausführen lässt.

Sie können die Aufgabenplanung verwenden, um ein Windows PowerShell-Skript zeitgesteuert zu starten. Der Befehl für die geplante Aufgabe ist *Powershell.exe*. Das Argument lautet `-command 'Pfad\Skript.ps1'`. Natürlich wird das Skript nur ausgeführt, wenn die Ausführungsrichtlinie dies zulässt.

Weitere Informationen Ausführen von Windows PowerShell-Skripts

Die folgenden Artikel bieten weitere Informationen über Windows PowerShell-Skripts und die Ausführungsrichtlinie:

- »Running Windows PowerShell Scripts« unter *http://go.microsoft.com/fwlink/?LinkID=192746*
- »Windows PowerShell – Abwehren von schädlichem Code« unter *http://technet.microsoft.com/de-de/magazine/2008.01.powershell.aspx*
- »Windows PowerShell – Hier signieren, bitte« unter *http://technet.microsoft.com/de-de/magazine/2008.04.powershell.aspx*
- »`Get-ExecutionPolicy`« unter *http://technet.microsoft.com/de-de/library/dd347644.aspx*
- »`Set-ExecutionPolicy`« unter *http://technet.microsoft.com/de-de/library/dd347628.aspx*

Lokale, globale und Remotebefehle

SharePoint-Cmdlets lassen sich in zwei Kategorien einteilen, nämlich in lokale und in globale Cmdlets. Lokale Cmdlets wirken nur auf einem einzigen Server. Beispielsweise verwenden Sie das Cmdlet `Start-SPServiceInstance`, um auf einem Server einen Dienst zu starten. Um einen neuen Server mit einer SharePoint-Farm zu verbinden, verwenden Sie das Cmdlet `Connect-SPConfigurationDatabase`. Wenn Sie einen Befehl auf mehreren Servern einer Farm

verwenden möchten, um auf diesen Servern zum Beispiel einen Dienst zu starten, müssen Sie den Befehl auf jedem Server separat zur Ausführung bringen.

Globale Cmdlets wirken sich auf die gesamte Farm aus, gewöhnlich durch entsprechende Änderungen in einer SQL Server-Datenbank. Wenn Sie zum Beispiel mit dem Cmdlet `Set-SPWebApplication` die Eigenschaften einer Webanwendung ändern, werden die Änderungen in die Inhaltsdatenbank übernommen und somit auf allen Servern wirksam, auf denen die Webanwendung verwendet wird. Sie brauchen die Server nicht alle einzeln zu konfigurieren. In vergleichbarer Weise ist eine Websitesammlung auf allen SharePoint-Servern verfügbar, wenn Sie mit `New-SPSite` eine neue Websitesammlung erstellen.

Das Windows PowerShell-Profil der SharePoint 2010-Verwaltungsshell

Die SharePoint 2010-Verwaltungsshell lädt ein Windows PowerShell-Profil aus dem SharePoint-Stammverzeichnis, standardmäßig: *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\CONFIG\POWERSHELL\Registration\SharePoint.ps1*. Ein Windows PowerShell-Profil ist ein Skript, das die Benutzerumgebung für Windows PowerShell einrichtet. Im Fall der SharePoint 2010-Verwaltungsshell führt das Profil drei wichtige Schritte durch:

- **Laden der SharePoint-Snap-Ins** Das SharePoint 2010-Verwaltungsshellprofil lädt die SharePoint-Snap-Ins.

  Wenn Sie die Windows PowerShell-Konsole statt der SharePoint 2010-Verwaltungsshell verwenden, können Sie keine SharePoint-Cmdlets verwenden, weil das SharePoint-Snap-In nicht vom Standardprofil geladen wird. Um das SharePoint-Snap-In zu laden, müssen Sie folgenden Befehl verwenden:

  ```
  Add-PSSnapin Microsoft.SharePoint.PowerShell
  ```

Hinweis Reflexion ist in SharePoint 2010 nicht mehr erforderlich

Ein anderer Weg, die gewünschte SharePoint-Funktionalität zu Windows PowerShell hinzuzufügen, führt über einen Prozess, der *Reflexion* genannt wird und bei dem die DLL-Dateien für SharePoint direkt geladen werden. Das war in SharePoint 2007 noch erforderlich, wird aber für SharePoint 2010 nicht empfohlen, weil nun das SharePoint-Snap-In verfügbar ist.

- **Hinzufügen des Stsadm-Pfads (*SharePoint-Stammverzeichnis/BIN*)** Das Profil fügt den Pfad zur Datei *Stsadm.exe* zur Umgebungsvariablen `Path` hinzu. Daher sind Sie nicht auf Windows PowerShell beschränkt, sondern können auch Stsadm verwenden.
- **Einstellen von *PSThread* auf *ReuseThread*** Diese Einstellung verbessert die Speichernutzung von Windows PowerShell und verringert die Wahrscheinlichkeit von Speicherlecks. In Windows PowerShell wird jede Zeile – jeder Befehl – mit ihrem eigenen Thread (oder Prozess) gestartet. Wenn `ThreadOptions` auf `ReuseThread` eingestellt ist, werden alle Befehle mit demselben Thread ausgeführt. Wenn Sie Windows PowerShell verwenden, müssen Sie folgenden Befehl verwenden:

  ```
  $Host.Runspace.ThreadOptions="ReuseThread"
  ```

Weitere Informationen PSThreadOptions

Der folgende Artikel bietet weitere Informationen über `PSThread`: »PSThreadOptions Enumeration« unter *http://go.microsoft.com/fwlink/?LinkId=183145*.

Berechtigungen für die Verwendung von Windows PowerShell

Damit ein Administrator Windows PowerShell für die Verwaltung von SharePoint 2010 verwenden kann, muss er für jede Datenbank, die mit Windows PowerShell verwendet werden soll, die Rolle *SharePoint_Shell_Access* haben. Um zum Beispiel Aufgaben durchführen zu können, bei denen Daten aus der Konfigurationsdatenbank gelesen oder geändert werden, muss einem Administrator die Rolle *SharePoint_Shell_Access* für diese Datenbank zugewiesen werden. In vergleichbarer Weise muss ein Administrator über die Rolle *SharePoint_Shell_Access* oder *db_owner* für die entsprechende Inhaltsdatenbank verfügen, wenn er mit einer bestimmten Websitesammlung arbeiten soll.

Um diese beiden Rollen zuzuweisen und somit die Berechtigung für die Verwendung von Windows PowerShell zu delegieren, können und sollten Sie das Cmdlet `Add-SPShellAdmin` verwenden. Der Vorgang ist sehr einfach.

Delegieren von Berechtigungen mit *Add-SPShellAdmin*

1. Öffnen Sie die SharePoint 2010-Verwaltungsshell.
2. Verwenden Sie das Cmdlet `Add-SPShellAdmin`, um einem Benutzer die Berechtigung zur Verwendung von Windows PowerShell für den Zugriff auf eine bestimmte Inhaltsdatenbank zuzuweisen. Orientieren Sie sich an folgendem Beispiel:

```
Add-SPShellAdmin -username <Domäne\Benutzer>
    -database (Get-SPContentDatabase <Name der Inhaltsdatenbank>)
```

Auf diese Weise können Sie einem Benutzer mit einem einzigen Befehl die Rolle *SharePoint_Shell_Access* zuweisen und den Benutzer auf jedem Server der Farm zur lokalen Gruppe *WSS_ADMIN_WPG* hinzufügen. Falls der Benutzer gerade angemeldet ist, muss er sich natürlich abmelden und wieder anmelden, damit die neue Mitgliedschaft in einer lokalen Gruppe wirksam wird.

Natürlich muss Ihr Konto über die Berechtigung verfügen, den Zugriff mit Windows PowerShell zu delegieren. Um `Add-SPShellAdmin` erfolgreich verwenden zu können, muss Ihr Konto über die Rolle *securityadmin* für die SQL Server-Instanz und die Rolle *db_owner* für die Datenbank verfügen, und Sie müssen auf jedem Server der Farm in der Gruppe *Administratoren* sein. Anders gesagt müssen Sie als Administrator über weitreichende Rechte verfügen, um einem anderen Benutzer die Berechtigung zur Verwendung von Windows PowerShell geben zu können. Aus praktischer Sicht sind Sie wahrscheinlich Administrator für SQL Server und jeden Server der Farm, aber aus rein technischer Sicht brauchen Sie eigentlich nicht ganz so viele Rechte.

Wenn Sie den Windows PowerShell-Zugriff auf eine Inhaltsdatenbank delegieren, delegieren Sie auch den Zugriff auf die Konfigurationsdatenbank. Das ist erforderlich. Es ist nicht möglich, nur den Zugriff auf eine Inhaltsdatenbank zu delegieren, ohne den Zugriff auf die Konfigurationsdatenbank zu delegieren.

Wenn Sie mit Windows PowerShell eine Inhaltsdatenbank erstellen, wird Ihrem Konto automatisch die Rolle *db_owner* für die Datenbank zugewiesen.

Außerdem müssen Sie ein Websitesammlungsbesitzer sein, wie er in der Zentraladministration definiert wird, um Windows PowerShell in der Inhaltsdatenbank für eine Websitesammlung verwenden zu können.

Das folgende Beispiel zeigt die Zuweisung eines Websitesammlungsbesitzers mit Windows PowerShell:

```
Set-SPSiteAdministration <WebsitesammlungsURL> -OwnerAlias <Domäne\Benutzer>
    -SecondaryOwnerAlias <Domäne\Benutzer>
```

Darin gilt:

- *<WebsitesammlungsURL>* ist die URL der Websitesammlung.
- Das Konto *<Domäne\Benutzer>* hinter dem Parameter `-OwnerAlias` gibt den primären Websitesammlungsadministrator an.
- Das Konto *<Domäne\Benutzer>* hinter dem Parameter `-SecondaryOwnerAlias` gibt den sekundären Websitesammlungsadministrator an.

Weitere Informationen Windows PowerShell

Die folgenden Quellen bieten ausführliche Informationen über Windows PowerShell und die Verwendung von Windows PowerShell zur Verwaltung von SharePoint:

- »Windows PowerShell« unter *http://go.microsoft.com/fwlink/?LinkID=192735*
- Das Windows PowerShell Script Center unter *http://go.microsoft.com/fwlink/?LinkID=192736*
- »Verwalten von SharePoint 2010-Produkten mit Windows PowerShell« unter *http://technet.microsoft.com/de-de/library/ee806878.aspx*

Schnelltest

- Können Sie SharePoint mit Windows PowerShell verwalten, wenn Sie Mitglied der Gruppe *Farm Administrators* sind?

Antwort zum Schnelltest

- Nein. Ihnen muss die Berechtigung zur Verwendung von Windows PowerShell zugewiesen werden. Diese Berechtigung lässt sich mit dem Cmdlet `Add-SPShellAdmin` zuweisen.

Übung Verwalten von SharePoint mit Windows PowerShell

Die Übungen leiten Sie bei wichtigen Arbeitsschritten an. Die Beschreibungen gehen an dieser Stelle allerdings nicht ins Detail. Gehen Sie sorgfältig vor und wenden Sie die Verfahren an, die in dieser Lektion und an anderen Stellen des Buchs beschrieben werden. Eine ausführlichere Anleitung finden Sie bei Bedarf im Abschnitt »Ausführliche Übungsanleitungen« am Ende dieses Buchs.

In dieser Übung erkunden Sie Windows PowerShell. Dann erstellen Sie eine Webanwendung, eine Websitesammlung und eine Website. Schließlich erstellen Sie Berichte, die Ihre SharePoint-Implementierung dokumentieren.

Vorbereiten der Übungen

Bevor Sie diese Übungen durchführen, müssen Sie eine Testumgebung aufbauen, wie in der Einführung dieses Buchs beschrieben. Außerdem müssen Sie die Übungen aus Lektion 1 dieses Kapitels durchgeführt haben.

▶ Übung 1 Verwenden bekannter Befehle in Windows PowerShell

In dieser Übung starten Sie die SharePoint 2010-Verwaltungsshell und verwenden Befehle, die Ihnen bereits bekannt sind.

1. Melden Sie sich als *CONTOSO\SP_Admin* auf SP2010-WFE1 an.
2. Starten Sie die SharePoint 2010-Verwaltungsshell mit der Option *Als Administrator ausführen*.
3. Geben Sie **dir** ein und drücken Sie die EINGABETASTE.
4. Verwenden Sie das Cmdlet `Get-Alias`, um die folgende Frage zu beantworten: Für welches Cmdlet ist `dir` ein Alias?
5. Geben Sie **ipconfig /all** ein und drücken Sie die EINGABETASTE.
6. Geben Sie **cls** ein und drücken Sie die EINGABETASTE.
7. Geben Sie **stsadm -help** ein.

▶ Übung 2 Delegieren der Berechtigung zur Verwendung von Windows PowerShell für die Verwaltung von SharePoint

In dieser Übung werden Sie feststellen, dass Sie nicht über die Berechtigung verfügen, eine vorhandene Inhaltsdatenbank mit Windows PowerShell zu verwalten. Dann gewähren Sie dem Konto *SP_Admin* die Berechtigung, die Datenbank mit Windows PowerShell zu verwalten.

1. Erstellen Sie einen Bericht über alle Websites einer Farm, einschließlich der Zentraladministration.

   Eine Fehlermeldung erscheint.

   Frage: Was geht aus der Fehlermeldung über die Ursache hervor? Wie können Sie das Problem beheben?
2. Beheben Sie das Problem, auf das Sie im vorigen Schritt gestoßen sind.

   Tipp: Um das Problem zu beheben, müssen Sie dem Konto *CONTOSO\SP_Admin* die erforderlichen Berechtigungen zuweisen. Zu diesem Zweck müssen Sie ein Windows PowerShell-Cmdlet mit höheren Rechten ausführen, in diesem Fall mit dem Konto *CONTOSO\Administrator* und dem Kennwort **Pa$$w0rd**.
3. Schließen Sie alle Instanzen der SharePoint 2010-Verwaltungsshell.
4. Starten Sie die SharePoint 2010-Verwaltungsshell als *SP_Admin* mit der Option *Als Administrator ausführen*.
5. Erstellen Sie einen Bericht über alle Websites einer Farm, einschließlich der Zentraladministration.

▶ Übung 3 Suchen und Untersuchen eines Windows PowerShell-Cmdlets

In dieser Übung suchen Sie den Befehl, mit dem sich eine neue Webanwendung erstellen lässt, und sehen sich dessen integrierte Dokumentation an.

1. Listen Sie die SharePoint-Cmdlets für Windows PowerShell auf.
2. Listen Sie die Cmdlets auf, die Aufgaben für Webanwendungen durchführen.

   Frage: Welches Substantiv steht für SharePoint-Webanwendungen?

   Frage: Welches Cmdlet erstellt eine SharePoint-Webanwendung?
3. Zeigen Sie die Kurzfassung der Dokumentation des Cmdlets `New-SPWebApplication` an.
4. Zeigen Sie die Anwendungsbeispiele für das Cmdlet an.

   Tipp: Sie können die NACH-OBEN-Taste verwenden, um einen zuvor eingegebenen Befehl erneut anzuzeigen, und den Befehl dann nach Bedarf überarbeiten.
5. Zeigen Sie den ausführlichen Hilfetext für das Cmdlet an.

   Frage: Welcher Parameter ist nur erforderlich, wenn es den Anwendungspool noch nicht gibt?

▶ Übung 4 Erstellen einer Webanwendung mit Windows PowerShell

In dieser Übung erstellen Sie mit Windows PowerShell eine Webanwendung, eine Websitesammlung und eine Stammwebsite.

1. Erstellen Sie eine Webanwendung namens *teams.contoso.com*. Verwenden Sie folgende Daten:
   - Name: Contoso Teams
   - Port: 80
   - Hostheader: *http://teams.contoso.com*
   - URL: *http://teams.contoso.com:80*
   - Anwendungspool: *SharePoint Web Applications*
   - Inhaltsdatenbankname: *SharePoint_Content_Teams*
2. Listen Sie die verfügbaren SharePoint-Websitevorlagen auf.

   Frage: Wie lautet der Name der *Teamwebsite*-Vorlage?
3. Erstellen Sie im Stamm der Webanwendung *teams.contoso.com* eine Websitesammlung. Verwenden Sie folgende Daten:
   - URL: *http://teams.contoso.com*
   - Inhaltsdatenbank für die Websitesammlung: *SharePoint_Content_Teams*
   - Name: Contoso Teams
   - Primärer Websitesammlungsadministrator: *CONTOSO\SP_Admin*
   - Vorlage: die *Teamwebsite*-Vorlage
4. Erstellen Sie in der Webanwendung *teams.contoso.com* eine Teamwebsite für die Abteilung *Finance*. Verwenden Sie folgende Daten:
   - URL: *http://teams.contoso.com/Finance*
   - Name: Finance
   - Vorlage: die *Teamwebsite*-Vorlage

5. Erstellen Sie in der Webanwendung *teams.contoso.com* eine Teamwebsite für die Abteilung *Marketing*. Geben Sie nur die ersten Buchstaben der Cmdlet- und Parameternamen ein und vervollständigen Sie die Namen mit der TABULATORTASTE. Verwenden Sie folgende Daten:
   - URL: *http://teams.contoso.com/Marketing*
   - Name: Marketing
   - Vorlage: `STS#0`
6. Erstellen Sie in der Webanwendung *teams.contoso.com* unter der Website *Marketing* eine neue *Blog*-Website. Verwenden Sie folgende Daten:
   - URL: *http://teams.contoso.com/Marketing/Blogs*
   - Name: Marketing Blogs
   - Vorlage: die *Blog*-Vorlage

▶ Übung 5 Erstellen von Berichten über Ihre SharePoint-Farm

In dieser Übung erstellen Sie Berichte über alle Websites in Ihrer SharePoint-Farm.

1. Listen Sie alle Websites in der Webanwendung *teams.contoso.com* auf. Erstellen Sie einen Befehl, der alle Websites in der Webanwendung anzeigt, auch wenn die Webanwendung viele Websitesammlungen und Websites enthält.
2. Weisen Sie die Website *Marketing Blogs* einer Variablen namens `$website` zu.
3. Listen Sie die Member eines `SPWeb`-Objekts auf. Verwenden Sie dafür die Variable `$website` und das Cmdlet `Get-Member`.
4. Listen Sie nur die Eigenschaften eines `SPWeb`-Objekts auf. Verwenden Sie wieder die Variable `$website` und das Cmdlet `Get-Member`.
5. Listen Sie alle Eigenschaften der Website *Marketing Blogs* und deren Werte auf. Verwenden Sie die Variable `$website` und das Cmdlet `Select-Object`.

   Frage: Welche Eigenschaft verrät, dass es sich bei der Website um einen Blog handelt?
6. Erstellen Sie einen Bericht über alle Websites einer Farm, einschließlich der Zentraladministration. Nehmen Sie die URL, die Vorlage und das Datum der letzten Aktualisierung der Website in den Bericht auf.

▶ Übung 6 Erstellen von Websites mit einem Windows PowerShell-Skript

In dieser Übung erstellen Sie ein Windows PowerShell-Skript für die Bereitstellung von Websites im Contoso-Intranet und verwenden es.

1. Erstellen Sie folgendes Skript:

```
$departments = ("HR","IT","Sales")
ForEach ($dept in $departments)
{
    New-SPWeb "http://teams.contoso.com/$dept" -Name "$dept" -Template "STS#0"
}
```

2. Speichern Sie das Skript unter dem Namen *CreateSites.ps1* auf dem Desktop.
3. Führen Sie das Skript in der SharePoint 2010-Verwaltungsshell aus.

   Die Websites werden erstellt.

4. Erstellen Sie einen Bericht über alle Websites in der Webanwendung *teams.contoso.com*.
5. Erstellen Sie einen Bericht über alle Websites in der Farm, einschließlich der Zentraladministration.

   Geben Sie dazu folgenden Befehl ein und drücken Sie die EINGABETASTE:

```
Get-SPWebApplication -IncludeCentralAdministration | Get-SPSite -Limit ALL |
    Get-SPWeb -Limit ALL
```

▶ Übung 7 Erstellen des Snapshots *CHAPTER 02*

Der Snapshot *CHAPTER 02* erfasst den Zustand der Umgebung am Ende von Kapitel 2. Führen Sie die Prozedur für folgende virtuelle Computer durch: SP2010-WFE1, CONTOSO-DC.

1. Fahren Sie den virtuellen Computer herunter.
2. Heben Sie die Bereitstellung des ISO-Abbilds im CD/DVD-Laufwerk auf. Orientieren Sie sich an der Beschreibung »Bereitstellung eines ISO-Abbilds aufheben« im Anhang »Anleitung zum Einrichten der Testumgebung«.
3. Erstellen Sie einen Snapshot namens *CHAPTER 02*. Orientieren Sie sich an der Beschreibung »Erstellen eines Snapshots« im Anhang »Anleitung zum Einrichten der Testumgebung«.

Zusammenfassung der Lektion

- Windows PowerShell ist das bevorzugte Befehlszeilentool für die Verwaltung und Automatisierung von SharePoint 2010.
- Die SharePoint 2010-Verwaltungsshell lädt die PowerShell-Cmdlets für Windows PowerShell automatisch, ist für die Verwaltung von SharePoint optimiert und unterstützt Stsadm.
- Windows PowerShell-Cmdlets folgen strengen Standards, beispielsweise der Syntax in der Form *»Verb-Substantiv«*, wobei die Substantive in einem Wort geschrieben werden. Auch die Dokumentation ist standardisiert.
- Mit den Cmdlets `Get-Command`, `Get-Help` und `Get-Member` können Sie Befehle suchen, deren Syntax und Parameter anzeigen und die Struktur der Objekte untersuchen.
- Windows PowerShell arbeitet mit .NET-Objekten. Ein Cmdlet gibt Objekte zurück und reicht sie in der Pipeline an das nachfolgende Cmdlet weiter. Objekte können auch zur späteren Verwendung in Variablen gespeichert werden.
- Das SharePoint-Objektmodell und somit auch die Windows PowerShell-Cmdlets verwenden eine Terminologie, die stark von der Terminologie in der Benutzeroberfläche abweichen kann.
- Variablen und Schleifen unterstützen die Erstellung von leistungsfähigen Skripts, die von Windows PowerShell ausgeführt werden können, sofern die geltende Ausführungsrichtlinie die Ausführung von Skripts zulässt.
- Um SharePoint mit Windows PowerShell verwalten zu können, müssen Ihnen die entsprechenden Berechtigungen zugewiesen werden.

Lernzielkontrolle

Mit den folgenden Fragen können Sie Ihr Wissen über den Stoff aus Lektion 2, »Automatisieren von SharePoint-Vorgängen mit Windows PowerShell«, überprüfen. Die Fragen finden Sie (in englischer Sprache) auch auf der Begleit-CD, Sie können sie also auch auf dem Computer im Rahmen eines Übungstests beantworten.

Hinweis Die Antworten

Die Antworten auf diese Fragen mit Erklärungen, warum die jeweiligen Auswahlmöglichkeiten richtig oder falsch sind, finden Sie im Abschnitt »Antworten« am Ende dieses Buchs.

1. Was kann das Cmdlet `Get-SPWebApplication` tun? (Wählen Sie alle zutreffenden Antworten.)
   A. Eine Website erstellen
   B. Ein Objekt zurückgeben, das eine einzelne Webanwendung repräsentiert
   C. Eine Auflistung von Objekten zurückgeben, die alle Webanwendungen in der Farm repräsentieren
   D. Eine Excel Services-Dienstanwendung erstellen
2. Sie möchten eine Websitesammlung löschen. Was können Sie verwenden?
   A. `Delete-SPSite`
   B. `Remove-SPWebApplication`
   C. `Remove-SPWeb`
   D. `Remove-SPSite`
3. Sie möchten einen Bericht über alle Websitesammlungen der Farm einschließlich der Zentraladministration erstellen. Was brauchen Sie? (Wählen Sie alle zutreffenden Antworten. Jede korrekte Antwort ist Teil der vollständigen Lösung.)
   A. Das Cmdlet `Get-SPWebApplication`
   B. Den Parameter `-IncludeCentralAdministration`
   C. Das Cmdlet `Get-SPSite`
   D. Den Parameter `-Limit ALL`

Rückblick auf dieses Kapitel

Um den in diesem Kapitel behandelten Stoff zu vertiefen und einzuüben, können Sie folgende Aufgaben durchführen:

- Lesen Sie die Zusammenfassung des Kapitels sorgfältig durch.
- Lesen Sie die Liste der Schlüsselbegriffe durch, die in diesem Kapitel eingeführt wurden.
- Arbeiten Sie die Übungen mit Fallbeispiel durch. Diese Szenarien beschreiben Fälle aus der Praxis, in denen die Themen dieses Kapitels zur Anwendung kommen. Sie werden aufgefordert, eine Lösung zu entwickeln.
- Arbeiten Sie die empfohlenen Übungen durch.
- Machen Sie einen Übungstest.

Zusammenfassung des Kapitels

- Die Verwaltung von SharePoint kann so aufgeteilt und delegiert werden, dass die betreffenden Benutzer nur die Berechtigungen und Rollen erhalten, die sie für die vorgesehenen Verwaltungsaufgaben brauchen.
- SharePoint lässt sich mit der Zentraladministration, mit Stsadm und mit Windows PowerShell verwalten.
- Durch die Verwendung von Windows PowerShell, dem bevorzugten Befehlszeilenwerkzeug für die Verwaltung und Automatisierung von SharePoint 2010, können Sie die meisten Verwaltungsaufgaben auf der Befehlszeile, interaktiv oder mit Skripts ausführen.
- Windows PowerShell ist sehr leistungsfähig, da es direkt mit den .NET-Objekten arbeitet, die SharePoint-Komponenten repräsentieren. Außerdem bietet Windows PowerShell zahlreiche Funktionen, mit denen sich die Cmdlets leicht finden, untersuchen und anwenden lassen, die Sie zur Durchführung von einfachen und komplizierten Aufgaben brauchen.

Schlüsselbegriffe

Die folgenden Begriffe wurden in diesem Kapitel neu eingeführt. Wissen Sie, was sie bedeuten?

- Alias
- Cmdlet
- Iterieren
- Member, Eigenschaften und Methoden
- Objekt
- Pipeline
- Remoting
- Variable

Übung mit Fallbeispiel: Erstellen von Berichten über die SharePoint-Farm

In der folgenden Übung mit Fallbeispiel wenden Sie Ihr Wissen über den Stoff dieses Kapitels an. Antworten auf die Fragen finden Sie im Abschnitt »Antworten« am Ende des Buchs.

Ihre Vorgesetzte hat einen wöchentlichen Bericht über die Websites in der Farm und über die Aktivität auf den Websites gefordert. Außerdem möchte sie wissen, bei welchen Websites es sich um Blogs handelt, damit sie überprüfen kann, ob das Material in den Blogs den Richtlinien von Contoso entspricht. Sie möchte die Berichte in einem Format, das sie mit Microsoft Excel sortieren und analysieren kann. Sie möchten die Berichte automatisch erstellen, damit Ihre Vorgesetzte die Berichte abrufen kann, ohne dass Sie dafür zusätzlich Zeit aufwenden müssen.

1. Mit welchen Windows PowerShell-Cmdlets können Sie eine Liste der Websites erstellen, die in der Farm vorhanden sind, und in der auch für jede Website die verwendete Vorlage und das Datum der letzten Änderung des Inhalts angeben wird?
2. Mit welchem Windows PowerShell-Cmdlet können Sie den Bericht in einem Format speichern, das eine Bearbeitung und Analyse in Microsoft Excel ermöglicht?
3. Welche Windows-Funktion können Sie einsetzen, damit jede Woche ein neuer Bericht erstellt wird?
4. Welche Anmeldeinformationen und Berechtigungen sind für diese Lösung erforderlich?

Empfohlene Übungen

Sie sollten die folgenden Aufgaben durcharbeiten, um die in diesem Kapitel behandelten Prüfungsziele einzuüben.

Übung 1: Untersuchen von Einschränkungen aus Sicherheitsgründen

In den Übungen von Lektion 1 haben Sie die Verwaltung von Dienstanwendungen, Websitesammlungen und Websites delegiert. Für die Benutzeroberfläche von SharePoint gelten gegebenenfalls Einschränkungen, die aus Sicherheitsgründen erfolgen. Verknüpfungen zu Aufgaben, für deren Durchführung keine Berechtigung vorliegt, werden entfernt. Melden Sie sich der Reihe nach als die Benutzer an, denen in den Übungen von Lektion 1 Berechtigungen zugewiesen wurden, und überprüfen Sie die Verwaltungsschnittstellen der Zentraladministration und der Seite *Websiteeinstellungen* der Contoso-Intranetwebsite *http://intranet.contoso.com*. Achten Sie darauf, welche Aufgaben für jeden Benutzer verfügbar sind und welche nicht.

Übung 2: Verwalten von Benutzerrollen mit Windows PowerShell

In Lektion 2 haben Sie sich mit der Verwendung von Windows PowerShell beschäftigt und Funktionen und Cmdlets kennengelernt, die eine weitere Einarbeitung in Windows PowerShell erleichtern. Verwenden Sie diese Funktionen und Cmdlets, wie `Get-Command`, `Get-Help` und `Get-Member`, um herauszufinden, wie Sie mit Windows PowerShell Verwaltungsaufgaben delegieren können. Wie können Sie einen Benutzer zur Gruppe *Farm Administrators* hinzufügen oder aus ihr entfernen? Wie können Sie einen Websitesammlungsbesitzer hinzufügen oder entfernen – die primären und sekundären Websitesammlungsadministratoren, wie sie

die Zentraladministration definiert? Wie fügen Sie einen Websitesammlungsadministrator zu einer Websitesammlung hinzu? Verwenden Sie nicht nur das Material aus diesem Kapitel, sondern suchen Sie auch im Microsoft TechNet und im Internet nach weiteren Informationen.

Machen Sie einen Übungstest

Die Übungstests (in englischer Sprache) auf der Begleit-CD zu diesem Buch bieten zahlreiche Möglichkeiten. Zum Beispiel können Sie einen Test machen, der ausschließlich die Themen aus einem Prüfungslernziel behandelt, oder Sie können sich selbst mit dem gesamten Inhalt der Prüfung 70-667 testen. Sie können den Test so konfigurieren, dass er dem Ablauf einer echten Prüfung entspricht, oder Sie können einen Lernmodus verwenden, in dem Sie sich nach dem Beantworten einer Frage jeweils sofort die richtige Antwort und Erklärungen ansehen können.

Weitere Informationen Übungstests

Einzelheiten zu allen Optionen, die bei den Übungstests zur Verfügung stehen, finden Sie im Abschnitt »So benutzen Sie die Übungstests« in der Einführung am Anfang dieses Buchs.