Einführung

In diesem Kapitel:

Diese Einführung enthält wichtige Informationen zu PowerShell und zum Gebrauch dieses Buchs, also sollten Sie diese vielleicht nicht überspringen.

Wer braucht eigentlich PowerShell?

In einer Welt, die eigentlich schon alles hat, ist es höchst ungewöhnlich, dass ein Unternehmen wie Microsoft zig Millionen US-Dollar in Entwicklungsarbeit investiert, um eine vollkommen neuartige Skript- und Automationssprache zu entwickeln: *Windows PowerShell*. Dass dies dennoch geschehen ist, hat mit Ihnen zu tun. Wieso genau, offenbart ein Blick auf PowerShell aus verschieden hohen Umlaufbahnen.

Moderne Lernkurve

Wer ganz nah um PowerShell kreist, vielleicht gerade selbst vor der PowerShell-Konsole sitzt, sieht vor allem Befehle, die Cmdlets. Sie funktionieren in etwa wie in anderen Konsolen (Shells). Allerdings unterstützt PowerShell so viele davon und ist auf so einheitliche Weise erweiterbar, dass man mit ganz geringen Grundkenntnissen fast alles damit administrieren kann. Noch wichtiger: Weil alle Cmdlets genau denselben Grundregeln folgen, kann man das Wissen auch leicht auf andere Aufgabenbereiche und Cmdlets anwenden.

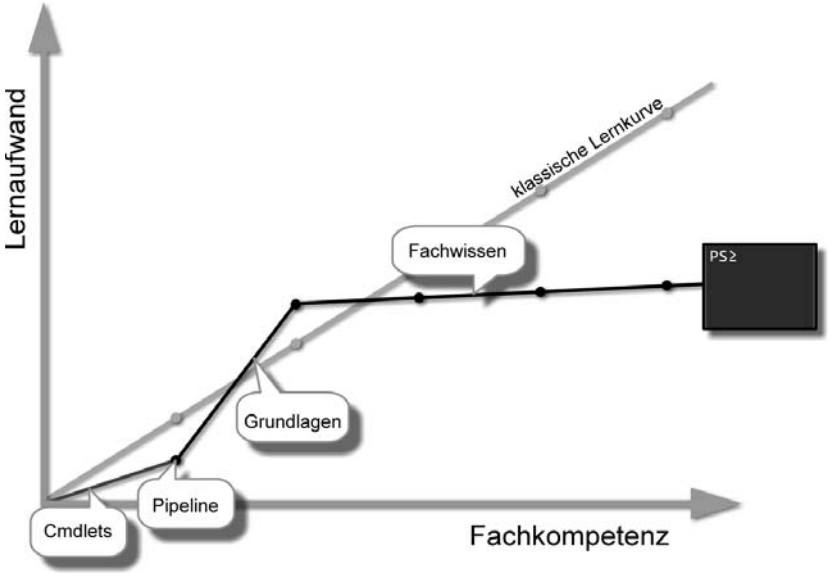

Abbildung E.1 Die PowerShell-Lernkurve ist nicht linear, sondern baut auf gemeinsamen Standards auf

PowerShell baut also auf Standards auf. Die ersten Schritte mit Cmdlets sind noch sehr einfach und benötigen kaum PowerShell-Kenntnisse. Auch der Umgang mit der PowerShell-Pipeline ist noch relativ einfach. Dann aber stößt man an eine Steilwand, die autodidaktisch oft nur schwer zu meistern ist. Hier werden die Grundlagen gelegt. Hier beschäftigt man sich mit Operatoren, Objekten und der wahren Natur von PowerShell. Genau diese Steilwand meistern Sie mit diesem Buch.

Sobald die Grundlagen einmal gelegt sind, flacht die Lernkurve stark ab, und alles wird gut: Ob Sie mit Microsoft Exchange ein Postfach anlegen, mit SharePoint eine Site veröffentlichen oder einfach nur Protokolldateien parsen oder Server verwalten wollen – Ihr PowerShell-Wissen befähigt Sie dazu, über den Tellerrand zu blicken und Ihre PowerShell-Erfahrung auch in ganz anderen IT-Bereichen einzusetzen. Entscheidend ist nun nur noch Ihr Fachwissen im jeweiligen Bereich.

Sie müssen also schon selbst wissen, *warum* Sie ein Exchange-Postfach anlegen oder eine Sharepoint-Website umbenennen wollen – *wie* das geschieht, ist aber dank der gemeinsamen PowerShell-Standards nun keine Frage mehr. *Get-ADUser*, *Get-Mailbox* und *Get-VM* beschaffen Ihnen mit denselben PowerShell-Grundkenntnissen Active Directory-Benutzerkonten, Exchange Server-Mailboxen oder VMware virtuelle Maschinen. Kennen Sie erst einmal ein Cmdlet, dann kennen Sie alle.

Computer – ich/ich – Computer: PowerShell als Fremdsprache

Entfernt man sich etwas von PowerShell und betrachtet die Technik mit mehr Abstand, verschwimmen die einzelnen Befehle zu etwas Neuem: einer Sprache. Tatsächlich benimmt sich PowerShell in vielen Aspekten genau wie eine Fremdsprache, wie Spanisch oder Italienisch etwa, nur dass Sie sich diesmal nicht mit dem Eisverkäufer unterhalten, sondern mit Ihrem Computer. Auch der Lernprozess, den Sie mit diesem Buch vor sich haben, verläuft ganz ähnlich.

Zuerst steht Vokabelpauken auf dem Plan, und schon nach wenigen Minuten werden Sie mit den ersten Vokabeln bereits einfache Aufgaben lösen. Tatsächlich werden einige Administratoren nie wirklich über diese Stufe hinauswachsen, weil es für sie vielleicht unökonomisch ist. Viele Urlauber kommen sehr gut mit ein paar Brocken Landessprache aus, um Wein, Käse und Baguette zu bestellen sowie nach Preis und Uhrzeit zu fragen. Nicht anders ist das in der IT. Hier zählt erst einmal nur, was essentiell ist, um eine Aufgabe zu lösen. Dieser erste Lernschritt bildet gleichzeitig den *Schwierigkeitsgrad 1* (**Einsteiger**) und bildet **Teil A** dieses Buchs (Kapitel 1–4).

Nach dem Vokabelpauken folgen Grammatik und Satzbau, und so ist das auch bei PowerShell. Sie lernen die PowerShell-Pipeline kennen, mit der Sie einzelne Wörter (Befehle) in geschliffene Ausdrücke verwandeln. Wer diese Grammatik beherrscht, kann plötzlich sehr viel differenziertere Aufgaben lösen und ist nicht mehr allein auf die Wortbrocken des Grundwortschatzes angewiesen. Auch hier gilt: Nicht jeder wird fließend PowerShell sprechen lernen, doch werden Sie auf jeden Fall fließend PowerShell verstehen lernen und dann die Ausdrücke anderer PowerShell-Sprechender mühelos nachvollziehen und für eigene Zwecke anpassen können. Der Satzbau entspricht *Schwierigkeitsgrad 2* (**Fortgeschrittene**) und bildet **Teil B** dieses Buchs mit den Kapiteln 5 bis 9.

Nachdem Sie die Sprache beherrschen, wird bei Ihnen mit einiger Wahrscheinlichkeit der Wunsch aufkommen, nicht nur Vorhandenes zu konsumieren, sondern auch zu produzieren. Sie werden eigene Funktionen und Skripts schreiben und als Module exportieren, also den Literaturmarkt von PowerShell vergrößern. Damit helfen Sie anderen, die auf einer anderen Stufe der PowerShell-Lernkurve stehen geblieben sind, denn Ihre Module sind im Grunde nichts weiter als neue Vokabeln, die auch von denjenigen rasch einsetzbar sind, die über das erste Vokabellernen nicht hinausgehen wollten. Alles zu diesem Thema entspricht *Schwierigkeitsgrad 3* (**PowerShell-Entwickler**) und findet sich in **Teil D** mit den Kapiteln 16 bis 22.

Der Olymp der PowerShell-Sprache ist natürlich die Poesie, bei der Sie mit den Nuancen der Sprache spielen, ihre unterschwelligen Stärken betonen und Lösungen schaffen, die über das hinauswachsen, was mit Alltagsausdrücken möglich ist. Nicht jeder wird Ihre Poesie verstehen, und manche PowerShell-Poesie liegt eigentlich im Grenzbereich zu anderen Sprachen und Lösungen. Dieses sanfte Grundverständnis für die Natur von PowerShell ist aber enorm wichtig, weil es auch bei ganz normalen und vollkommen unesoterischen Aufgabenstellungen den Kitt bildet, um mit den vorhandenen Befehlen die gesuchte Lösung zu erstellen. Deshalb ist dieses Thema zwischen Satzbau und Skriptbau bereits in **Teil C** mit den Kapiteln 10 bis 15 untergebracht. Der *Schwierigkeitslevel 4* (**PowerShell-Profi**) ist für viele möglicherweise etwas schwere Kost, doch muss man dieses Kapitel nicht von vorn bis hinten durchlesen oder gar auswendig kennen. Es genügt zu wissen, dass es diese Techniken gibt, falls man auf normalem Wege einmal nicht weiterkommt.

Schließlich finden Sie in **Teil E** mit den Kapiteln 23 bis 28 noch ganz spezielle erwähnenswerte PowerShell-Techniken wie Remotezugriffe oder Hintergrundjobs, die für Benutzer aller Schwierigkeitsgrade gleichermaßen spannend und interessant sind.

Eine strategische Plattform

Entfernt man sich noch etwas weiter von PowerShell, rücken plötzlich andere Prioritäten in den Vordergrund und PowerShell erscheint wiederum in neuem Licht. PowerShell ist lösungsorientiert, ja, aber es ist auch eine strategische Plattform, deren Bedeutung andere Automationssprachen weit überstrahlt. Tatsächlich ist PowerShell eine neuartige zweite Benutzeroberfläche, wenn auch so andersartig als die gewohnte Maus-Fenster-Klick-Oberfläche, dass man dies zuerst gar nicht wahrnimmt. Microsoft hat nicht nur eine Skript- und Automationssprache entwickelt, um VBScript ins .NET-Zeitalter zu überführen oder die antiquierte Befehlszeile zu renovieren, sondern um eine langfristige Lösung zu schaffen für ein Problem, das viele heute vielleicht noch gar nicht wahrnehmen:

Unsere IT-Landschaft wird immer komplexer und muss sich immer höheren Anforderungen stellen, ökonomischen genau wie regulatorischen. Gleichzeitig steht nicht zu erwarten, dass qualifiziertes Fachpersonal in gleichem Maße aus dem Boden sprießt wie die Nachfrage danach steigen wird. Administratoren müssen also vielseitiger werden. Versuchen Sie heutzutage, einen SQL Server-Spezialisten mit der Administration von Exchange zu betrauen, dürfte die Katastrophe absehbar sein. Da aber beide Produkte über PowerShell verwaltbar sind, ist es mit PowerShell sehr viel leichter, auch in anderen IT-Bereichen Verantwortung zu übernehmen.

Hier werden die Parallelen zur grafischen Benutzeroberfläche besonders deutlich: Während man Maus, Fenster und Menüs als Selbstverständlichkeit intuitiv bedient und sich auf die Eigenarten der Programme konzentriert, gilt Ähnliches für PowerShell: Der Einsatz der Sprache selbst – der Cmdlets, Pipeline, Erweiterungsmodule – gehört zu den Selbstverständlichkeiten, über die man bald nicht mehr bewusst nachdenkt.

Administratoren müssen nicht nur vielseitiger werden, sondern sich auch die Arbeit erleichtern, um die nötigen Freiräume dafür zu schaffen. Das bei Kuckucksuhren und Schokoladenkonfekt begehrte Prädikat »handgemacht« verliert in der IT langsam, aber sicher seinen Charme. Eine durchgehend manuelle Administration kostet zu viel Zeit, ist zu fehleranfällig und entspricht nicht modernen juristischen und regulatorischen Dokumentationsansprüchen.

Da die Aufgaben in der IT aber vielerorts noch nicht einmal annäherungsweise standardisiert sind, werden viele Aufgaben nur mit maßgeschneiderten Skripts automatisierbar sein. Auch hier bildet PowerShell einen ökonomischen Standard. PowerShell-Skripts sind für alle IT-Bereiche einsetzbar, lösen also auf Wunsch das bunte Sammelsurium aus Perl, VBScript, KiXtart, Stapeldateien (Eingabeaufforderung) etc. ab, konsolidieren also die Automation. PowerShell-Skripts sind einheitlich dokumentierbar, signierbar und integriert in die Standardsicherheitsmechanismen von Windows.

Nicht zu unterschätzen ist dabei die Modularisierbarkeit. Während früher Berater und externe Firmen Skriptauftragslösungen erstellt haben, welche die Aufgaben zwar erledigten, aber kaum verständlich, nachvollziehbar, wartbar oder weiterverwertbar waren, bestehen solche Auftragslösungen bei PowerShell aus Erweiterungsmodulen mit den gewünschten neuen »Vokabeln«. Der eigentliche Workflow ist davon getrennt, und Ihr Unternehmen kann diese Vokabeln künftig auch in ganz anderem Zusammenhang nutzen, um mit dem Computer zu »sprechen« und Aufgaben zu lösen.

Schließlich lebt PowerShell längst nicht nur in der interaktiven PowerShell-Konsole, sondern steht als direkte Programmierschnittstelle zur Verfügung. Viele Softwarehersteller nutzen PowerShell bereits als interne Skriptsprache, und künftig wird die Automationsoberfläche PowerShell immer stärker mit der grafischen Oberfläche Windows verschmelzen. Schon heute liefern viele Dialogfelder und Assistenten zum Schluss den PowerShell-Code, der das bewerkstelligt, was Sie im Dialogfeld zusammengeklickt haben.

Abbildung E.2 verdeutlicht nicht nur den PowerShell-Zyklus, sondern bildet auch den roten Faden durch dieses Buch. PowerShell ist in diesem Buch in drei große Abschnitte eingeteilt:

- **Einsteiger** Sie haben noch nie von PowerShell gehört und/oder wollen ohne Programmierung schnell Ergebnisse erzielen
- **Fortgeschritten** Sie möchten mehr Einfluss haben und sich nicht mit dem abfinden, was Cmdlets Ihnen liefern, und setzen deshalb die PowerShell-Pipeline ein, um Befehlsergebnisse Schritt für Schritt in die richtige Form zu bringen
- **PowerShell-Entwickler** Sie wollen selbst neue Befehle und sogar eigene Module schreiben, und Sie wollen direkt auf die Low-Level-Funktionen des Betriebssystems zugreifen

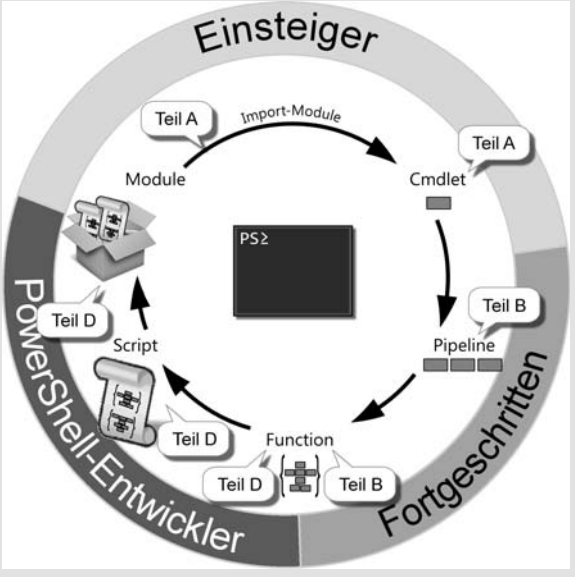

Abbildung E.2 PowerShell-Zyklus: Selbst komplexe Skripts werden über Module wieder zu einfachen Cmdlets

Der Einsteigerbereich grenzt links direkt an den (sehr viel komplexeren) PowerShell-Entwicklerbereich: Während es ganz einfach ist, vorhandene Module zu importieren, um neue PowerShell-Befehle zu nutzen, ist das Erstellen ganz neuer eigener Module nicht so einfach und liegt im Reich der PowerShell-Entwickler.

Auch der Fortgeschrittenen-Bereich stößt an den PowerShell-Entwicklerbereich an: Wohingegen es für einen fortgeschrittenen PowerShell-Anwender leicht ist, Abläufe als eigene Funktionen zu gestalten, bedarf es noch etwas mehr Fachwissen, um aus diesen kleinen Funktionen echte und vollwertige Skript-Cmdlets zu machen.

Die Übergänge sind also fließend und sie sind offen: Sie können jederzeit von einem Bereich in einen anderen wechseln und Abbildung E.2 zeigt jeweils, welcher Buchteil den entsprechenden Teil des PowerShell-Zyklus behandelt.

Plattformübergreifende Technik

Entfernt man sich noch einmal ein Stück weiter von PowerShell, treten plötzlich auch andersartige Systeme in den Blickwinkel. Was Sie nun verschwommen sehen, ist zwar heute noch weitgehend Vision, aber die ersten Grundzüge der Umsetzung sind bereits erkennbar. Während PowerShell zurzeit noch eine reine Windows-Technik ist, wird es damit mittelfristig auch möglich, plattformübergreifend andere Systeme zu verwalten und einzubinden. Schon heute ist dies mit einer Technik namens *WSMan* möglich.

Persönliche Entwicklung

Schließlich kann PowerShell auch eine Karriereentscheidung sein. Wann immer Sie eine Aufgabe lösen, stehen dahinter unsichtbare Motivationen. Die offensichtlichste ist natürlich, die Aufgabe gut zu erledigen, denn dafür werden Sie (wahrscheinlich) bezahlt. Ebenso wichtig ist aber auch, was die Lösung dieser Aufgabe sonst noch für Sie bedeutet und wie sie in Ihre Lebensbilanz einfließt. Wer sich Tag für Tag durch Dialogfelder klickt, kann zwar enorm erfolgreich Aufgaben lösen, entwickelt seine Fähigkeiten aber nicht weiter, und wenn das Dialogfeld eines Tages nicht mehr da ist, gilt das vielleicht auch für den eigenen Arbeitsplatz. Zwar wird es immer gute Gründe für Klicklösungen geben, aber sobald Sie eine Aufgabe mehr als einmal durchführen müssen, sollten Sie über PowerShell nachdenken.

Es mag Sie anfangs etwas mehr Zeit kosten, die Lösung damit zu automatisieren, aber bedenken Sie: Jede Extraminute, die Sie hier investieren, investieren Sie eigentlich in Ihre persönliche Fortbildung. Auch ein Arbeitgeber sollte dies als Chance verstehen und Freiräume dafür gestatten. Denn mit jeder erfolgreich gemeisterten PowerShell-Lösung wächst Ihre Sprachfertigkeit. Wer PowerShell am Ende fließend spricht, ist allerbestens aufgestellt für moderne IT-Landschaften. Falls doch mal etwas schiefgeht, mag es Sie trösten, dass das wiederholte Schlagen mit der Stirn auf die Schreibtischoberfläche pro Stunde immerhin 68 Kalorien verbraucht.

Gerade falls Sie vorher noch nie geskriptet haben, sehen Sie PowerShell als Chance: Wer den Zug vielleicht zu VBScript-Zeiten vor zehn Jahren verpasst hat und sich nun etwas abgehängt vorkommt, kann heute auf einen neuen Zug aufspringen. Mit diesem Buch haben Sie alles, was Sie wissen müssen, und können sich natürlich auch zurückgezogen im stillen Kämmerlein und bei eigenem Tempo in PowerShell einarbeiten – um dann plötzlich und unerwartet als neuer Skriptguru das Rampenlicht zu betreten.

Wie Sie dieses Buch nutzen

Dieses Buch ist in mehrere Teile gegliedert, von denen Sie ja schon gehört haben. Es setzt keinerlei Grundkenntnisse voraus, zumindest wenn Sie von vorn beginnen zu lesen. Wer unter Zeitdruck steht, kann aber auch quer einsteigen, und wer noch weniger Zeit hat, findet in jedem Kapitel Zusammenfassungen, in denen die jeweils wichtigsten Inhalte für Krisenzeiten zusammengefasst sind.

Die PowerShell-Beispiele im Buch sind jeweils in einer anderen Schriftart formatiert. Damit Sie leichter erkennen, welche Eingaben von Ihnen erwartet werden, wird bei allen Eingaben die PowerShell-Eingabeaufforderung »PS>« vorangestellt. Diese Eingabeaufforderung kann bei Ihnen auch anders aussehen und sollte in den Beispielen natürlich nicht mit eingegeben werden.

Viele PowerShell-Codebeispiele sind sehr kurz und können mit geringem Aufwand schnell eingetippt werden. Umfangreichere Beispiele sind mit einer Listingunterschrift gekennzeichnet. Unter dem dort genannten Dateinamen finden Sie die Codebeispiele auch in den Begleitmaterialien, die Sie hier herunterladen können: *http://www.microsoft-press.de/support/9783866456877* oder *http://msp.oreilly.de/support/2262/767*.

Noch mehr Unterstützung

Falls bei der Arbeit mit diesem Buch Fragen auftauchen oder Sie Anregungen haben, besuchen Sie mich: *http://www.powertheshell.com*. Oder senden Sie mir eine Nachricht an *tobias.weltner@email.de*.

Dieses Buch ist stark geformt worden durch meine jahrelange Arbeit als PowerShell-Trainer für Unternehmen im Mittelstand und Großkundensegment. An dieser Stelle möchte ich mich bei dieser Gruppe bedanken: den vielen Administratoren, Projektleitern, Consultants und anderen IT-Profis, die mich mit ihren vielen äußerst konkreten und praxisnahen Fragen immer wieder erden und dafür sorgen, dass dieses Buch sehr praxisnah geraten ist.

Abbildung E.3 Ein großes Dankeschön an meine Kursteilnehmer für viele gute Ideen und Anregungen!

Falls Sie ebenfalls Interesse haben, an einem meiner PowerShell-Trainings teilzunehmen, beispielsweise einem Bootcamp, einem Umsteigerseminar oder bei einer Inhouseveranstaltung direkt vor Ort, dann freue ich mich über Ihre E-Mail. Sie erreichen mich für Anfragen und Details unter *tobias.weltner@email.de*.

Bevor ich Ihnen nun endlich viel Spaß mit PowerShell wünsche, geht noch ein großes Dankeschön an meinen Fachlektor Thomas Irlbeck, der dieses Buch mit allergrößtem Sachverstand und mit größter Sorgfalt lektoriert hat. Seine zahlreichen guten Anregungen und Ergänzungen sind ein wichtiger Teil dieses Buchs geworden.

Herzlichst Ihr
Dr. Tobias Weltner