Skriptsprachen-Abschlussprojekt

„Importieren und Löschen von Kontakten, Dokumenten und Feiertagen in Microsoft Sharepoint mittels der Powershell“

von

Thomas Kramer

Aufgabenstellung

Importierung von …

• iCal-Kalendereinträgen.
• Personeneinträgen.
• Dokumenten.

Importziel ist Microsoft Sharepoint, das Importieren soll mittels Powershell-Skripten geschehen.

Pflichtenheft

• Importierung aller Datensätze
• Loggen aller Fehlermeldungen in eine Logdatei
• Separate Skripte für Reversibilität, jeweils Importieren und Löschen soll möglich sein
• Richtige Zeichensatzkodierung nach Import der Datensätze sichtbar
• korrekte Benutzung des deutschen Datumsformats bei Kalendereinträgen
• korrekte Schreibweise der Postleitzahlen bei Personeneinträgen
• Alle Datensätze werden auch angezeigt
• Verknüpfungen in Webseite sind nach Import richtig
• Testen der Export-Schnittstelle, ob die Daten genauso exportiert werden wie sie importiert wurden

Testumgebung

Die Testumgebung stellt eine virtuelle Maschine im VirtualBox-Format dar, welche unter der akademischen MSDN-AA-Lizenz lizenzierte Versionen von Windows Server 2003 und Sharepoint Server 2007 enthält.

Ein relevanter Punkt war dass der Arbeitsrechner ein Notebook mit einem Pentium DualCore T4200@2 GHZ und 4 GB RAM darstellt. Auf diesem läuft zwar ein 64-bittiges Windows 7, aber der Prozessor stellt einen abgespeckten Core2Duo ohne Intels Hardware-Virtualisierung VT-x dar.

Es ist tatsächlich so dass Hardware-Virtualisierung benötigt wird um auf einem beliebigen Rechner eine 64-Bit-VM zu starten - ob der Wirt ein 32- oder 64-Bit-Betriebssystem benutzt ist dabei irrelevant (zumindest bei VirtualBox). Und die neuere Sharepoint-Version 2010 wird von Microsoft nur noch als 64-Bit-Version zur Verfügung gestellt.

Das bedeutet dass für die VM nicht die neue Version 2010 sondern nur die ältere Version 2007 von Sharepoint benutzt werden konnte. Eine wesentliche Einschränkung stellt das aber nicht dar.

Weiterhin enthalten ist die Windows Powershell 2.0, welche von Microsoft kostenfrei heruntergeladen werden kann – diese wird als Schnittstelle benutzt um die Daten importieren zu können.

Die aktuell installierte Version der Powershell kann mit folgendem PS-Befehl abgefragt werden:

```
(get-item
"hklm:\SOFTWARE\Microsoft\PowerShell\1\PowerShellEngine").GetValue("PowerShellVersion")
```

Sharepoint

Microsoft Sharepoint stellt eine Groupware-Lösung von Microsoft für das Intranet dar, demnach eine spezialisierte Software für die gemeinsame Arbeit in Gruppen. Das System ist modular aufgebaut und frei konfigurierbar – es sind etliche Module enthalten die eingebunden und zur Verfügung gestellt werden können, aber nicht **müssen**.

Folgende Funktionen werden u. a. zur Verfügung gestellt:

- Dokumentenbibliothek
- Kalender
- Wiki
- Kontakte
- Aufgaben
- Teamdiskussionen

Sharepoint stellt eine Webbrowser-basierte Lösung dar, das bedeutet es muss auf den Clients keine separate Software installiert werden.

Sämtliche Funktionen sind über den Webbrowser verfügbar; aber es gibt natürlich dennoch eine Benutzerverwaltung. Wenn nicht anders eingerichtet muss sich daher zum Aufrufen der Webseite oder für administrative Zwecke zunächst eingeloggt werden.

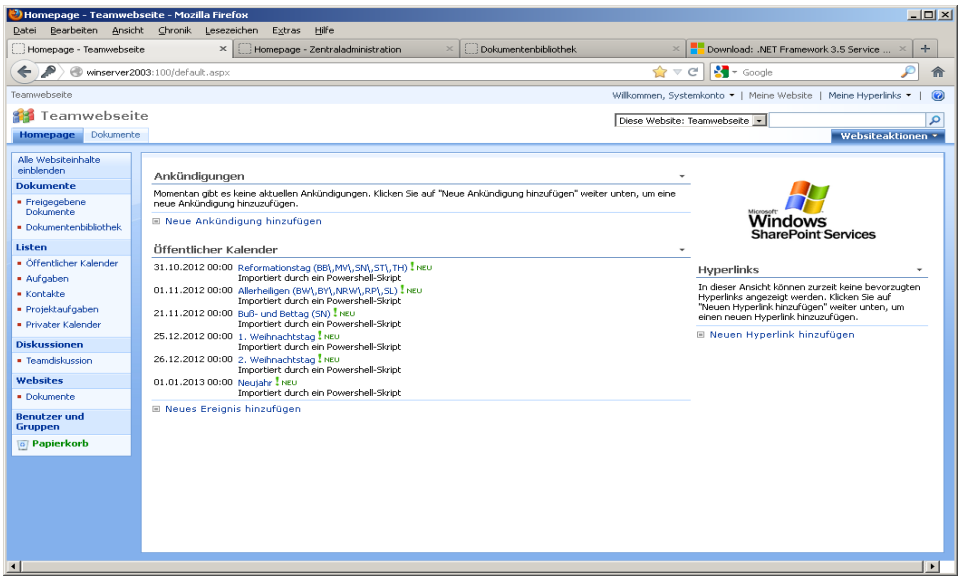
Die Webanwendung wird im Webbrowser Firefox mit der URL

http://localhost:100/

aufgerufen. Wenn das Login-Fenster kommt bitte als Benutzer **administrator** und als Passwort **thomas** eingeben. Sharepoint richtet sich bei seinen Benutzern nach den Windows-Accounts, die übernommen werden können.

Das Passwort gilt demnach auch für den Administrator-Account von Windows.

Nach dem Einloggen sieht man folgendes Fenster:



Die als Importziel verwendeten Funktionen von Sharepoint sind über die Leiste links erreichbar, namentlich sind dies: „Kontakte“, „Öffentlicher Kalender“ und „Dokumentenbibliothek“.

Sharepoint besitzt auch eine Administrationswebseite die sich Zentraladministration nennt und über die URL

http://localhost:14195

erreichbar ist. Diese wird aber für das Nachvollziehen der Import-Skripte nicht mehr benötigt.

Sharepoint unterscheidet weiterhin zwischen Webseitensammlungen und Webseiten. Wenn man als Administrator eine neue Webanwendung einrichtet wird zunächst eine neue Webseitensammlung gemäß einem auswählbaren Template eingerichtet. Anders als man vermuten könnte sind die in dem verwendeten Template enthaltenen Unterfunktionen aber keine eigenen Webseiten.

Das ist insofern relevant als dass beim Importieren mittels Sharepoint eine Webseitensammlung ausgewählt werden **muss** und eine Webseite ausgewählt werden **kann**. In dieser VM wurde testweise die zusätzliche Webseite „Dokumente“ eingerichtet, welche aber nicht weiter benutzt wird. Bitte nicht die Karteireiter oben zur Auswahl der Webseiten, sondern nur die Links in der Leiste links beachten.

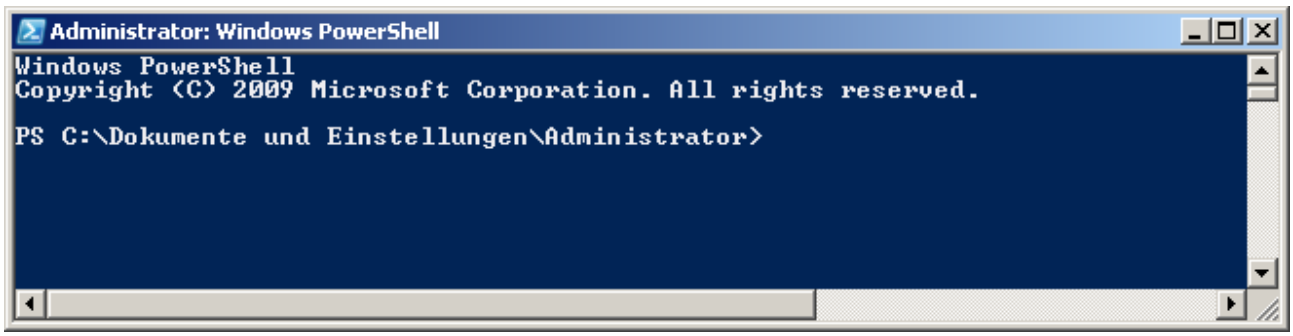
Windows Powershell

Die Windows Powershell ist in der Version 2.0 enthalten. In der VM ist sowohl die direkte Powershell als auch der Debugger **Powershell ISE** enthalten; zum Testen wurde eher letztere verwendet.

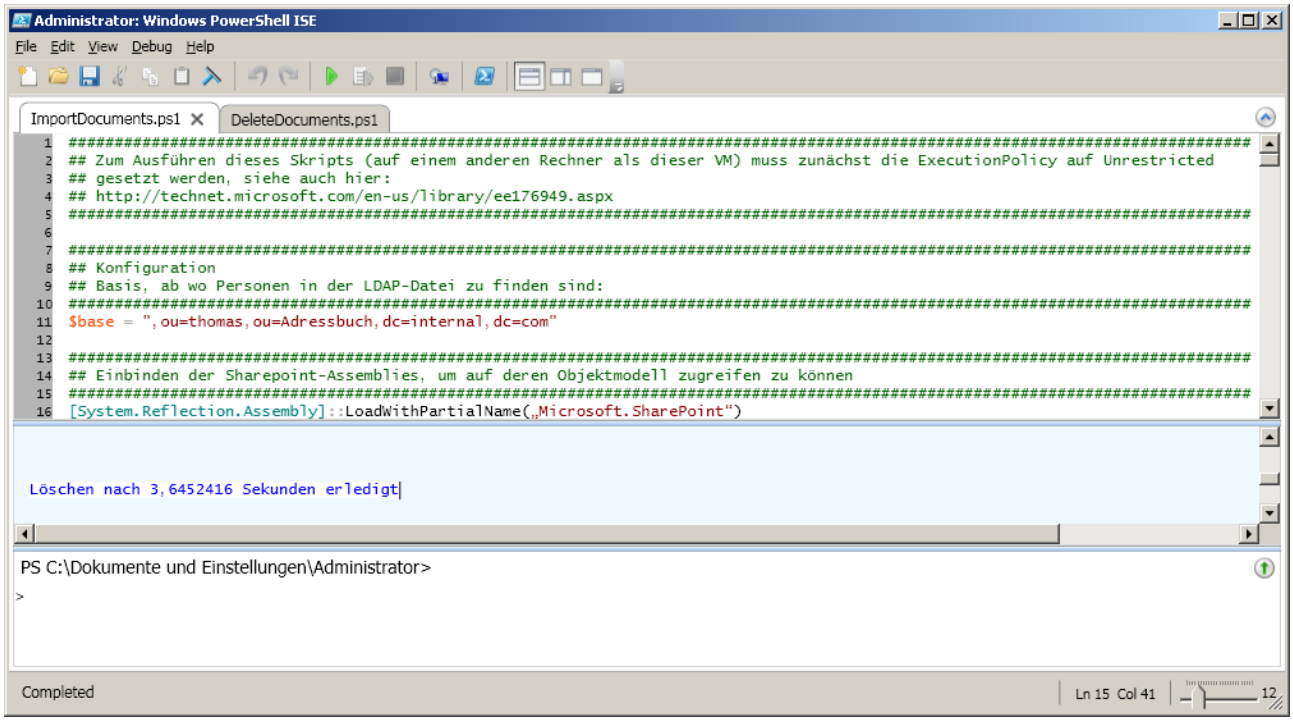
Die geschriebenen Skripte liegen in dem Powershell-Ordner, der auf dem Desktop des Administrator-Benutzers liegt (nach Einloggen direkt zu sehen). Die Skripte tragen die Namen **ImportDocuments.ps1** und **DeleteDocuments.ps1**. Powershell-Skripte haben also die Dateiendung .ps1.

Wenn die Powershell gestartet wird hat sie als Standardeinstellung den Pfad „C:\Dokumente und Einstellungen\Administrator\“ voreingestellt, diesen bitte beibehalten weil die Skripte über relative Pfade auf Unterordner zugreifen.

Nachfolgend eine Abbildung der Powershell [1]:



Nachfolgend eine Abbildung des Powershell ISE-Debuggers [2]:



Die Aufteilung des Debuggers [2] ist ganz einfach: Oben wird der Quellcode editiert, in der Mitte erfolgt die Ausgabe der Powershell und unten können Befehle direkt eingegeben und getestet werden.

Wenn die Skripte direkt mit der Powershell [1] aufgerufen werden sollen sollten sie mit den folgenden Befehlen gestartet werden:

Ohne Umlenkung der Fehlerausgabe:
./Desktop/Powershell/ImportDocuments.ps1
./Desktop/Powershell/DeleteDocuments.ps1

Mit Umlenkung der Fehlerausgabe:
./Desktop/Powershell/ImportDocuments.ps1 2> ./Desktop/Powershell/Errorlog.txt
./Desktop/Powershell/DeleteDocuments.ps1 2> ./Desktop/Powershell/Errorlog.txt

Die 2 gibt den Kanal an (Fehlermeldungen) und mit > wird die Log-Datei zunächst gelöscht und dann neu geschrieben. Mit >> würde stattdessen angehängt werden.

Im Powershell ISE-Debugger können die Skripte dagegen einfach über die Menüs geladen werden. Oben ist ein Icon zum Ausführen der Skripte vorhanden – besonders beachten bitte auch das Scheibenwischer-Symbol, welche das siebte Icon von links darstellt. Diese Funktion leert das Ausgabefenster.

Es geht also um zwei verschiedene Skripte; eines importiert Kontakte, Dokumente und Feiertage und ein anderes löscht die Daten wieder.

Was importiert wird

Folgende Daten werden mit dem Powershell-Skript ImportDocuments.ps1 importiert:

- 1. Personeneinträge mit eMail-Adressen
- 2. Jeweils ein zugehöriges Textdokument
- 3. Die Feiertage 2012.

Die Personeneinträge liegen in dem LDIF-Format vor, welches von (neueren) LDAP-Versionen zum Informationsaustausch verwendet wird.

LDAP ist eine zentrale Datenbank in Unternehmen und wird z. B. für Adressbücher und Benutzerdatenbanken, aber auch für Single-Sign-Ons verwendet damit man sich für verschiedene Dienste nur einmalig anmelden muss. Active Directory von Microsoft basiert ebenfalls auf LDAP.

Das Import-Skript liest die Personen aus der Datei Personen.ldif aus und importiert sie als Kontakte in Sharepoint. Wichtig ist als Einstiegspunkt in dem Skript die Variable $base, die den LDAP-Einstiegspunkt kennzeichnet von welcher Hierarchie-Ebene importiert werden soll. Man muss aber nichts verändern, mit den Voreinstellungen funktioniert der Import bereits.

Wenn eine Person gefunden und importiert wurde wird nach einem Dokument mit dem Dateinamen „Nachname, Vorname.txt“ in dem Unterordner Dokumente gesucht und ebenfalls importiert. Zunächst sollte ein Passfoto als Datei für jeden Kontakt verwendet werden, aber wegen der Rechte wurde dann jeweils eine Textdatei aus dem Gutenberg-Projekt verwendet welches alte Bücher digitalisiert zur freien Verfügung stellt.

Abschließend wird eine iCal-Kalenderdatei mit den Feiertagen für 2012 importiert. Hierbei war übrigens wichtig dass als Ende-Datum in ics-Dateien jeweils der darauffolgende Tag angegeben wird - aber das ist für den Kalender-Import natürlich falsch, es muss Ende-Datum – 1 Tag gerechnet werden. Um an dieser Stelle auf selbst implementierte und fehleranfällige Datumsrechnung zu verzichten wurde in der Powershell nach DateTime gecastet, dann -1 gerechnet und anschließend wieder ausgelesen.

Alle importierten Kalender- und Kontakte-Datensätze werden mit einer GUID (ID) gekennzeichnet um sie von händisch angelegten Datensätzen zu unterscheiden. Diese GUID wird in einer benutzerdefinierten Spalte gespeichert, die händisch hinzugefügt wurde.

Die Einstellungen wurden so verändert dass nur der Administrator importierte Kalender- und Kontakte-Datensätze bearbeiten kann - die anderen Benutzer sehen diese Datensätze zwar, können sie aber nicht verändern. In Sharepoint gibt es zwar für benutzerdefinierte Ansichten Zeilen/Datensatz-Filter aber keine Spalten-Filter, so dass sich das Anzeigen der scriptGUID-Spalte nicht vermeiden lässt.

Weiterhin bitte auch die Quellcode-Kommentare in den Skripten beachten.

Das Skript DeleteDocuments.ps1 löscht alle importierten Daten wieder, wobei darauf geachtet wird nur importierte Datensätze mit der speziellen GUID-Kennzeichnung zu entfernen.

Export-Schnittstelle der Powershell

Die Export-Schnittstelle, die mit dem Befehl **Export-SPWeb** gemäß der Microsoft-Seite

http://technet.microsoft.com/de-de/library/ee428301.aspx

benutzt wird konnte nicht getestet werden da die Fehlermeldung

„The term 'Export-SPWeb' is not recognized as the name of a cmdlet, function, script file, or operable program.“

erfolgte.

Besondere Schwierigkeiten

Einige Schwierigkeiten lassen sich anhand der Quellcode-Kommentare ablesen, aber auf eines sollte gesondert hingewiesen werden.

Der Powershell-Debugger zeigt keine kontextabhängige Informationen zu einem Objekt an. Wenn man in Visual Studio nach einem Objektnamen einen Punkt eingibt werden alle Methoden etc. des Objekts in einer Liste aufgelistet, aus der man auswählen kann. In der Powershell ISE gibt es so etwas nicht.

Entsprechend war es nicht einfach herauszufinden wie die Felder von Sharepoint intern eigentlich heißen. Hierzu empfehle ich das folgende Skript:

```
[System.Reflection.Assembly]::LoadWithPartialName(„Microsoft.SharePoint“)
$site = new-object -typename Microsoft.SharePoint.SPSite(„http://localhost:100/“)
$web = $site.OpenWeb()
$web.lists
```

Ein Variablenname ohne eine Zuweisung oder einer sonstige Aktion listet in der Powershell einfach sämtlichen Inhalt auf. Die Auflistung die durch den Befehl **$web.lists** erfolgt ist anschließend sehr lang, folgende Zeilen sind relevant:

```
Title                : Kontakte
BaseTemplate         : Contacts
SchemaXml            : …..
```

Untereinander sind damit alle Listen der Sharepoint-Seite aufgelistet. Über den folgenden Befehl

```
$list = $web.lists[„Kontakte“]
```

wird ein spezieller Bereich ausgewählt, den man ansprechen möchte – z. B. für den Import von Kontakten. Anders als man meinen könnte spricht man ihn aber über keinen internen Namen an, sondern über den, der auch dem Anwender erscheint (in der Liste als „Title“ gekennzeichnet):

```
$list = $web.lists[„Öffentlicher Kalender“]
```

ist also durchaus zulässig. Alternativ könnte man auch $web.lists.[Nummer] verwenden, aber das wäre möglicherweise nicht dauerhaft in dieser Reihenfolge.

Jedenfalls wurde folgende Vorgehensweise verwendet: Wenn Kontakte importiert werden sollten wurde zunächst in der Webanwendung Neuanlegen angeklickt und nachgeschaut wie die Felder mit deutschem Titel heißen. Dann wurde in der oben erwähnten Ausgabe nach dem Block gesucht der

```
Title                : Kontakte
```

beinhaltet und in diesem Block wurde in der SchemaXML-Eigenschaft nachgeschaut.

Die sichtbaren Feldbezeichnungen tauchen dort als DisplayName-Eigenschaft auf und davor erscheint eine Name-Eigenschaft, die den internen Kennzeichner darstellt über den das Feld angesprochen werden muss.

Referenzen

Hilfreich sind:

Windows Powershell: Einführung und Beispiele
http://w8.xn--apfelbck-s4a.de/782-windows-powershell-einfuhrung-und-beispiele

SharePoint-Zugriff mit der Powershell
http://www.searchnetworking.de/themenbereiche/administration/client-server-administration/articles/138682/

Windows Powershell v2.0 Reference
http://blogs.technet.com/b/csps/p/psref.aspx

Skriptressourcen für IT-Experten
http://gallery.technet.microsoft.com/scriptcenter/site/search?f%5B0%5D.Type=ScriptLanguage&f%5B0%5D.Value=PowerShell&f%5B0%5D.Text=PowerShell

Scripting Guy-Blog
http://blogs.technet.com/b/heyscriptingguy/