Informations
Organisations
Zentrum

Ausbildung Lernende

Lerndokumentation von Thomas Gassmann PowerShell

Autor
Gassmann Thomas

Version
1.0.0.0

Letzte Revision
25.11.2019 22:55

Dokumentstatus
Draft

Microsoft Partner
Gold Collaboration and Content
Silver Application Development
Silver Midmarket Solution Provider
Cloud Accelerate

IOZ Informations Organisations Zentrum AG · St. Georg-Strasse 2a · 6210 Sursee
Telefon 041 925 84 00 · Fax 041 925 84 01 · info@ioz.ch · www.ioz.ch

Contents

1.1 Beginning Powershell Scripting for Developers

1.1.1 Strings Formatieren

In PowerShell kann man Strings wie auch in C# mit einer String.Format Methode formatieren. Diese kann folgendermassen aufgerufen werden:

[string]::Format(„{0}“, $Variable)

In PowerShell geht dies aber etwas einfacher. Man kann zuerst den String angeben, danach –f und die Variablen auflisten.

„{0}{1}“ –f $Var1, $Var2

Dabei kann man auch die Variablen direkt formatieren:

{0:N0}	erste Variable: N für Zahl, keine Nachkommastellen
{0:N1}	erste Variable, N für Formatierung einer Zahl, 1 Nachkommastelle
{0,8:N0}	erste Variable, acht Leerzeichen vor der Zahl, formatiert als Zahl, keine Nachkommastellen
{0:C0}	erste Variable, als Währung formatiert, keine Nachkommastellen
{0:P0}	erste Variable, als Prozent formatiert, keine Nachkommastellen
{0:X0}	erste Variable, als HexDec Zahl formatiert, keine Nachkommastellen
{0:D0}	Dezimalzahl
{0:M/d/yyyy}	Datum formatiert

Auch Wildcards sind möglich:

```
# Wildcards
Clear-Host
"PowerShell" -like "Power*"
"PowerShell" -like "arcane*"
"PowerShell" -like "?owerShell"  # question marks work for single characters
"PowerShell" -like "Power*[s-v]" # ends in a char between s and v
"PowerShell" -like "Power*[a-c]" # ends in a char between a and c
```

Im folgenden Beispiel wird eine Telefonnummer überprüft. Dabei wird gesagt das die ersten 3 Stellen von 0-9 sein müssen, danach ein Strich kommen soll und die nächsten 3 Stellen wieder von 0-9 sein müssen:

```
# Regular Expressions
Clear-Host
"888-368-1240" -match "[0-9]{3}-[0-9]{3}-[0-9]{4}"
"ZZZ-368-1240" -match "[0-9]{3}-[0-9]{3}-[0-9]{4}"
"888.368.1240" -match "[0-9]{3}-[0-9]{3}-[0-9]{4}"
```

1.1.2 Arrays

Beim Initialisieren eines Arrays in PowerShell kann man die einzelnen Elemente des Arrays mit einem Komma trennen:

$array = „1“, „2“

Durch das Verwenden der Variable Array ohne einen spezifischen Index anzugeben, werden alle Elemente aufgelistet.

In PowerShell ist es jedoch speziell, dass man mit += neue Elemente zum Array hinzufügen kann.

Ausserdem kann man eine Reihe von Elementen hinzufügen durch das Verwenden von „..“. Dies kann dann beispielsweise so aussehen:

$array = 1..5

Mit –contains kann man überprüfen, ob ein Element im Array vorhanden ist. Es gibt aber auch –notcontains.

1.1.3 Multidimensionale Arrays

Um Multidimensionale Arrays in PowerShell zu erstellen muss man zuerst die einzelnen Arrays erstellen und diese dann zu einem anderen Array hinzufügen.

```
# Load four individual arrays
$a = 1..5
$b = 6..10
$c = 11..15
$d = 16..20

# Now create an array from the four individual ones
$array = $a, $b, $c, $d
```

```powershell
# Take the contents of the array and join them into a single string.
$array[0] -join " "
```

1.1.4 Hashtables

Hashtables in PowerShell können wie Dictionarys in C# verwendet werden.

```powershell
#-------------------------------------------------------------------------#
# Hash tables
#-------------------------------------------------------------------------#

$hash = @{"Key"         = "Value";
          "PowerShell"  = "PowerShell.com";
          "Arcane Code" = "arcanecode.com"}

$hash                   # Display all values
$hash["PowerShell"]     # Get a single value from the key

$hash."Arcane Code"     # Get single value using object syntax
```

Auf Hashtables gibt es auch zahlreiche Methoden wie Remove, Contains oder ContainsValue.

1.1.5 Branching

If, Elseif und Else funktionieren in PowerShell sehr ähnlich wie in C#. Der grösste Unterschied dabei ist aber dass man nicht Operatoren wie ==, != oder > verwenden kann. Man muss statt == beispielsweise –eq verwenden.

Auch Switch-Case funktioniert in PowerShell sehr ähnlich wie in C#. Dies funktioniert in PowerShell auch mit Collections. Bei Switch-Case sind strings Case-Insensitive und es spielt keine Rolle ob diese gross oder klein geschrieben werden. Man kann aber „switch –casesensitive“ benutzen. Mit „switch –Wildcard“ kann man auch Wildcards in den Abfragen bei Switch verwenden.

```powershell
$var = 42
switch ($var)
{
   41 {"Forty One"}
   42 {"Forty Two"}
   43 {"Forty Three"}
   default {"default"}
}
```

Mit „;“ kann man in PowerShell mehrere Befehle auf einer Linie haben.

1.1.6 Loops

In PowerShell gibt es neben do while auch do until. Das führt einen Loop solange aus, bis die Bedingung true ist.

1.1.7 $? Operator

Wenn der Command vor dem $? Operator einen Error hatte, wird "false" zurückgegeben, ansonsten "true".

```
[xml]$config = Get-Content ($PSScriptRoot + "\Config.xml")

if ($? -eq $false) {
    Write-host "Cannot load configuration source XML." "Error"
    return $null
}
return $config
```