Przewodnik po Windows PowerShell

Dostęp do argumentów

Aby uzyskać dostęp do argumentów linii poleceń, które podaje się przy starcie skryptu, należy wykorzystać zmienną **$args**. Można przeglądać zawartość zmiennej **$args** korzystając z polecenia:

```
foreach ($i in $args) {$i}
```

Aby uzyskać dostęp do określonego argumentu, należy skorzystać z numeru (indeksu) argumentu. Argumenty numerowane są od zera, więc 0 odpowiada za pierwszy argument w zbiorze argumentów, 1 odpowiada za drugi itp.:

```
$args[0]
```

Aby odnieść się do ostatniego argumentu na liście należy skorzystać z polecenia: `$args[-1]`

Zmiana kolorów tekstu oraz tła

Aby wyświetlić tekst w kolorze należy skorzystać z polecenia **Write-Host** i określić kolor :

```
Write-Host "test" -foregroundcolor "green"
```

Można również określić kolor tła:

```
Write-Host "test" -backgroundcolor "red"
```

Wstawianie podziału wiersza

Aby wstawić podział wiersza do ciągu wynikowego (*ang. output*) należy użyć znaków: **`n**:

```
Write-Host "Linia 1.`nLinia 2."
```

Pisanie w negatywie

Aby w trybie tekstowym wyświetlić wiadomość w negatywie, należy skorzystać z polecenia **Write-Warning**:

```
Write-Warning "Wystąpił błąd."
```

Wstawianie komentarzy

Aby wstawić komentarz, należy użyć znaku '**#**':

```
# To jest komentarz, a nie polecenie.
```

Pobranie danych wejściowych

Aby pobrać ciąg znaków od użytkownika, należy skorzystać z polecenia: **Read-Host**.

```
$a = Read-Host "Please enter your name"
```

Wstawianie pustej linii

Aby wstawić pustą linię w skrypcie PowerShell należy skorzystać ze znaku odwróconego apostrofu (`):

```
Write-Host `
    "To jest kontynuacja linii."
```

Można także "złamać" linię, wstawiając separator "|" między poszczególnymi poleceniami:

```
Get-ChildItem C:\Scripts |
    Sort-Object Length -Descending
```

Linie składające się z wielu poleceń

Aby wstawić wiele poleceń do jednej linii, można je odseparować znakiem średnika ";":

```
$a = 1,2,3,4,5; $b = $a[2]; Write-Host $b
```

Porównywanie

Polecenia PowerShell (np. takie jak **Where-Object**) wykorzystują operatory porównania, włączając w to te pokazane w tabeli.

–lt	Mniejsze
–le	Mniejsze lub równe
–gt	Większe
–ge	Większe lub równe
–eq	Równe
–ne	Nierówne
–like	Podobne (używa symboli wieloznacznych, *ang. wildcards*)
–notlike	Niepodobne (używa symboli wieloznacznych, *ang. wildcards*)

Każdy z operatorów może rozpoznawać wielkie i małe litery (*ang. Case-sensitive*). W tym celu należy dodać znak **c** przed operatorem. Na przykład **–ceq** to operator równości z uwzględnieniem wielkości znaków. Operator **–clt** to operator mniejszości z uwzględnieniem wielkości znaków.

Przewodnik po Windows PowerShell

Czytanie z pliku tekstowego

Aby przeczytać zawartość pliku tekstowego należy skorzystać z polecenia **Get-Content**, podając jako parametr ścieżkę do pliku:

```
$a = Get-Content C:\Scripts\Test.txt
```

Każda linia w pliku posiada swój numer. Numer ten zapisany jest w tablicy $a[<numer>]. Aby odnieść się do poszczególnych linii należy określić ich numer (indeks):

```
$a[0]
```

Aby uzyskać dostęp do ostatniej linii:

```
$a[-1]
```

Aby określić liczbę linii, słów, znaków w pliku tekstowym należy skorzystać z polecenia:

```
Get-Content c:\scripts\test.txt |
Measure-Object -line -word -character
```

Zapisywanie do pliku tekstowego

Aby zapisać dane w pliku tekstowym należy skorzystać z polecenia: **Out-File**:

```
Get-Process | Out-File C:\Scripts\Test.txt
```

Aby dołączyć dane do pliku, który już posiada zawartość należy skorzystać z parametru **–append**:

```
Get-Process | Out-File C:\Test.txt -append
```

Można także skorzystać z tzw. strumieni, znanych z systemu MS-DOS: '>' do zwykłego zapisu, albo '>>' do dopisania do istniejącej zawartości, np.:

```
Get-Process > C:\Scripts\Test.txt
```

Polecenie **Export-CSV** zapisuje dane w formacie CSV (rozdzielając je przecinkami):

```
Get-Process | Export-CSV C:\Test.csv
```

Drukowanie danych

Aby wydrukować dane na domyślnej drukarce, należy skorzystać z polecenia **Out-Printer**:

```
Get-Process | Out-Printer
```

Tworzenie instrukcji warunkowych

Aby utworzyć instrukcję warunkową "jeżeli" (if) należy skorzystać ze schematu:

```
$a = "white"
if ($a -eq "red")
    {"The color is red."}
elseif ($a -eq "white")
    {"The color is white."}
else
    {"The color is blue."}
```

Zamiast pisać wiele instrukcji warunkowych można skorzystać z instrukcji "Switch":

```
$a = 2
switch ($a)
    {
        1 {"Kolor czerwony."}
        2 {"Kolor niebieski."}
        3 {"Kolor zielony."}
        4 {"Kolor żółty."}
        default {"Inny."}
    }
```

Tworzenie pętli "For" i "For Each"

Aby utworzyć pętlę "For" należy skorzystać ze schematu:

```
for ($a = 1; $a -le 10; $a++) {$a}
```

Dla porównania, pętla "For Each" będzie wyglądała tak:

```
foreach ($i in get-childitem c:\scripts)
{$i.extension}
```

Tworzenie pętli "While" i "Until"

Aby utworzyć pętlę "While" lub "Until", należy skorzystać z podanych dwóch przykładów kodu, zamieniając kod w nawiasach {…} oraz warunek pętli, na kod, który ma zostać wykonany w pętli.

```
$a = 1
do {$a; $a++}
while ($a -lt 10)

$a = 1
do {$a; $a++}
until ($a -gt 10)
```

Przewodnik po Windows PowerShell

Tworzenie obiektów COM

Aby rozpocząć pracę z obiektami COM, należy skorzystać z polecenia **New-Object** z parametrem **–comobject** podając identyfikator programu tzw. ProgID:

```
$a = New-Object -comobject `
    "Excel.Application"
$a.Visible = $True
```

Dostęp do obiektów .NET

Aby skorzystać z obiektów .NET Framework, należy nazwę klasy zawrzeć w nawiasach kwadratowych. Następnie należy oddzielić nawę klasy i metodę używając do tego pary dwukropków:

```
[system.Net.DNS]::resolve("207.46.198.30")
```

Aby utworzyć odniesienie do obiektu w .NET Framework, należy skorzystać z polecenia **New-Object**:

```
$a = new-object `
-type system.diagnostics.eventlog `
-argumentlist system
```

Uwaga. To jest tylko ogólny zarys tego jak pracować z .NET. Pokazane w przykładzie techniki nie stosują się do wszystkich klas w .NET Framework.

Wyświetlanie właściwości

Aby uzyskać dostęp do właściwości obiektu, należy przekierować wyjściowy strumień danych do polecenia **Select-Object**:

```
Get-Process | Select-Object Name, Company
```

Sortowanie danych

Aby posortować dane zwracane w PowerShell, należy przekierować wyjściowy strumień danych do polecenia **Sort-Object** określając kryterium, zgodnie z którym chce się dane posortować:

```
Get-Process | Sort-Object ID
```

Można dodać parametry **–descending** lub **–ascending** aby określić kolejność sortowania:

```
Get-Process | Sort-Object ID -descending
```

Można posortować uwzględniając wiele właściwości:

```
Get-Process | Sort-Object ProcessName, ID
```

Praca z WMI

Aby uzyskać informacje o komputerze przy wykorzystaniu WMI, należy skorzystać z polecenia **Get-WMIObject** podając jako parametr nazwę klasy:

```
Get-WMIObject Win32_BIOS
```

Jeżeli klasa nie występuje w obszarze nazw *cimv2*, należy skorzystać z parametru **–namespace**:

```
Get-WMIObject SystemRestore `
    -namespace root\default
```

Aby uzyskać dostęp do danych na innym komputerze, należy skorzystać z parametru **–computername**:

```
Get-WMIObject Win32_BIOS `
    -computername atl-ws-01
```

Aby ograniczyć zwracane dane, należy skorzystać z zapytania WQL i parametru **–query**:

```
Get-WMIObject -query `
    "Select * From Win32_Service `
        Where State = 'Stopped'"
```

Budowanie dowiązań do Active Directory

Aby utworzyć dowiązanie do konta w Active Directory, należy skorzystać z dostawcy LDAP:

```
$a = [adsi] "LDAP://cn=kenmyer, `
    ou=Finance, dc=fabrikam, dc=com"
```

Wyliczanie obiektów w OU jest nieco bardziej skomplikowane. Istnieje jednak metoda na rozwiązanie tego zadania: utworzenie dowiązania do OU, następnie skorzystanie z metody **PSBase_GetChildren()** w celu otrzymania zbioru elementów zawartych w OU:

```
$objOU = [ADSI]`
"LDAP://ou=Finance,dc=fabrikam,dc=com"
$users = $objOU.PSBase.Get_Children()
$users | Select-Object displayName
```

Budowanie dowiązań do kont lokalnych

Aby utworzyć dowiązanie do konta lokalnego należy skorzystać z dostawcy WinNT:

```
$a = [adsi] "WinNT://atl-ws-01/kenmyer"
$a.FullName
```

Przewodnik po Windows PowerShell

Uzyskiwanie pomocy

Aby uzyskać dostęp do pomocy PowerShell należy skorzystać z polecenia **Get-Help** z parametrem **–full**. Przykładowo, aby uzyskać informacje dotyczące polecenia "Get-Process", należy posłużyć się poleceniem:

```
Get-Help Get-Process -full
```

Aby zobaczyć przykłady wykorzystania wybranego polecenia należy skorzystać z parametru **–examples**:

```
Get-Help Get-Process -examples
```

Można również skorzystać z polecenia **Get-Command**, aby wyświetlić listę dostępnych poleceń PowerShell:

```
Get-Command
```

Aby uzyskać listę dostępnych aliasów należy skorzystać z polecenia **Get-Alias**:

```
Get-Alias
```

Zmiana ustawień zabezpieczeń

Aby móc uruchamiać skrypty PowerShell, należy zmienić ustawienia zabezpieczeń. Domyślnie, w PowerShell mogą być uruchamiane jedynie te skrypty, które zostały podpisane cyfrowo przez zaufanego wystawcę certyfikatu. Aby umożliwić uruchamianie skryptów, które zostały utworzone lokalnie (niezależnie od tego, czy zostały podpisane cyfrowo, czy nie) należy użyć polecenia:

```
Set-ExecutionPolicy RemoteSigned
```

Zaglądanie do obiektu

Aby uzyskać informacje o właściwościach i metodach obiektu, należy uzyskać dostęp do instancji tego obiektu, a następnie przekierować obiekt do polecenia **Get-Member**. Przykładowo, poniższe polecenie zwraca właściwości i metody dostępne przy pracy z procesami:

```
Get-Process | Get-Member
```

Czyszczenie okna konsoli

Aby wyczyścić zawartość okna PowerShell należy skorzystać z polecenia **Clear-Host** (lub aliasu: **cls**).

Kopiowanie / Wklejanie

Aby włączyć możliwość prostego kopiowania i wklejania do konsoli Windows PowerShell, należy wykonać następujące kroki:

1. Uruchomić Windows PowerShell, kliknąć prawym przyciskiem myszy ikonę w lewym górnym rogu, wybrać **Properties**.
2. W oknie **Windows PowerShell Properties** na zakładce **Options** wybrać **QuickEdit Mode**, kliknąć OK.

Aby skopiować tekst z konsoli PowerShell należy zaznaczyć tekst i nacisnąć Enter. Aby wkleić tekst do konsoli należy kliknąć w obszarze konsoli prawym przyciskiem myszy.

Uruchamianie skryptów

Aby uruchomić skrypt z konsoli Windows PowerShell, należy wpisać pełną ścieżkę do skryptu (lub podać nazwę skryptu, jeśli skrypt znajduje się w bieżącej lokalizacji):

```
C:\Scripts\Test.ps1
```

Jeśli nazwa ścieżki zawiera spacje należy pełną nazwę ująć w znakach cudzysłowu. Na przykład:

```
&"C:\Scripts\My Scripts\test.ps1"
```

Aby uruchomić skrypt nie uruchamiając bezpośrednio konsoli Windows PowerShell (np. z Menu Start lub z konsoli cmd.exe), należy przywołać konsolę Windows PowerShell, podając ścieżkę do skryptu oraz argument wywołania:

```
powershell.exe -noexit C:\Scripts\Test.ps1
```

Parametr **–noexit** powoduje pozostawienie okna PowerShell na ekranie po wykonaniu się skryptu.

Więcej informacji

Aby uzyskać więcej informacji o pisaniu skryptów Windows PowerShell, odwiedź stronę Technet Script Center: http://www.mcrosoft.com/technet/scriptcenter/hubs/msh.mspx.
Swoją przygodę zacznij z webcast'ami *PowerShell Week* oraz serią artykułów: *A Task-Based Introduction to Windows PowerShell.*