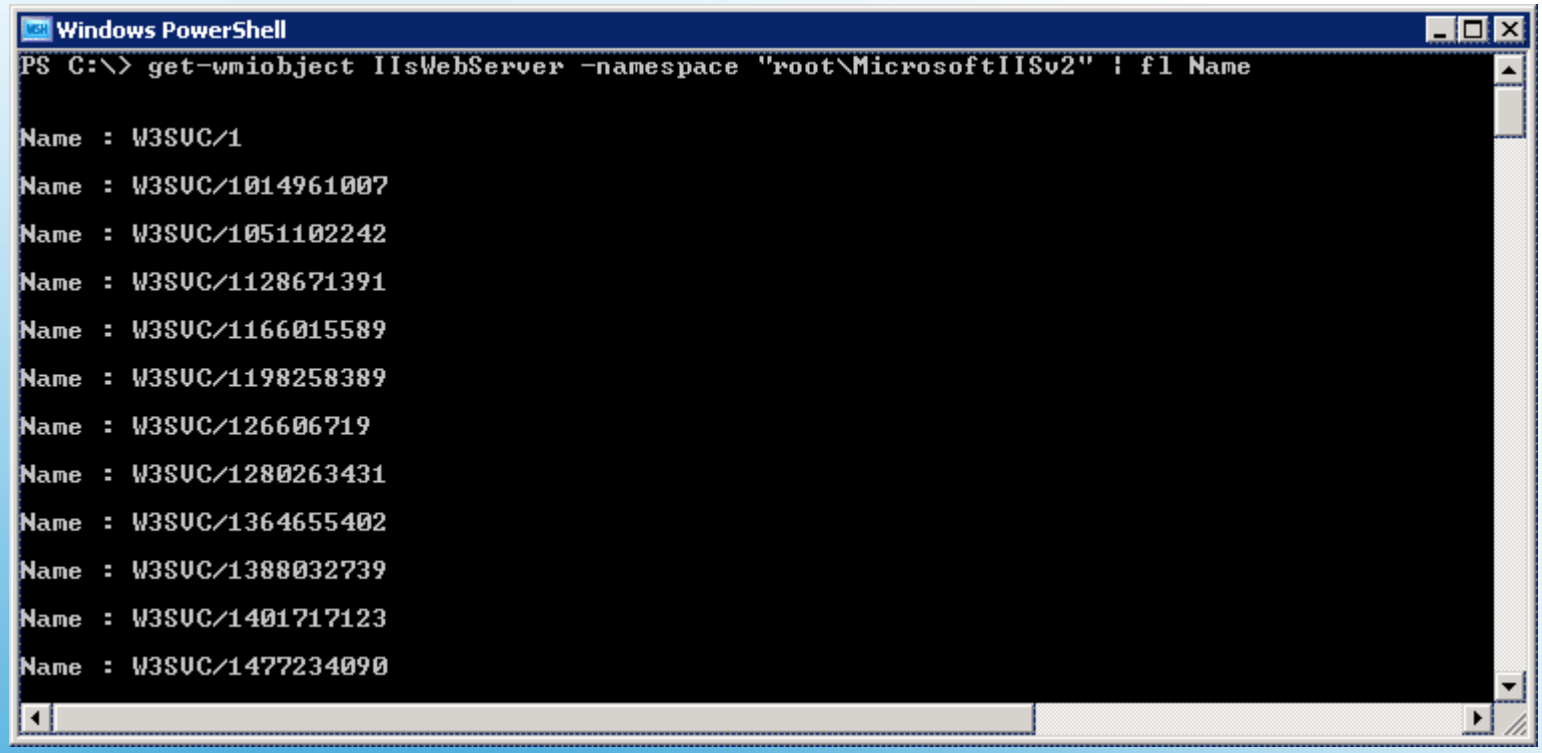
PowerShell



Sławomir Wawrzyniak

05.11.2010

Czym jest PowerShell

- Czym jest PowerShell
- Do czego może się przydać
- Zalety PowerShell
- Podobieństwo do basha

Wprowadzenie

- Jak uruchomić PowerShell

- Główne okno PowerShell

Nawigacja po systemie plików

Komendy nawigacyjne takie jak w systemie Dos
i w większości powłok linux.

dir	Listuje zawartość katalogu
pwd	Wyświetla aktualne położenie
cd	Przejście do katalogu
.	Odnosi się do bieżacego katalogu
..	Odnosi się do katalogu “wyżej”

Uruchamianie narzędzi

Z poziomu PowerShell możemy uruchamiać inne narzędzia np.

- ipconfig

- ping

- notepad

Polecenia ustrukturyzowane (cmdlety)

- Jedna z głównych różnic między cmd i PowerShell

- Wszystkie cmdlety maja nazwy w formacie Czasownik-rzeczownik np. Get-Process

- Nie trzeba wpisywać pełnych nazw cmdlet, można użyć klawisza tab w celu automatycznego uzupełnienia komend.

Obiekty

Najprostszym sposobem stworzenia stringa w PowerShell jest wpisanie “jakis tekst” do lini komend W zwiazku z tym ze nic nie przechwytuje stringa PowerShell go wyświetli.

Tak wygenerowany string jest pełnowartościowym obiektem .NET Freamwork I możemy swobodnie odwoływać się do jego właściwości.

Wszystkie polecenia, które zwracają dane wyściowe zwracają je również w postaci obiektów.

Zmienne

W PowerShell nazwy zmiennych zaczynają się od znaku $

Zmienne

W PowerShell nazwy zmiennych zaczynają się od znaku $

Przykład:
$proces = Get-Process notepad

Zmienne

W PowerShell nazwy zmiennych zaczynają się od znaku $

Przykład:
$proces = Get-Process notepad

Wywołanie metody:
$proces.Kill()

Stałe

PowerShell obsługuje podstawowe stałe administracyjne takie jak MB I GB

Stałe

PowerShell obsługuje podstawowe stałe administracyjne takie jak MB I GB

Zadanie:
Ile plyt cd potrzeba do zapisu 40 GB danych ?

Stałe

PowerShell obsługuje podstawowe stałe administracyjne takie jak MB I GB

Zadanie:
Ile plyt cd potrzeba do zapisu 40 GB danych ?

PS> 40GB / 700MB

Łączenie poleceń

Jeżeli polecenie generuje dane wyjściowe można użyć znaku potoku “|” aby przekazać te dane do innego polecenia.

Przyklad:

PS> Get-Item path* | Move-Item -Destination path2

Jak chronić się przed sobą

Jeżeli nie jesteśmy w stanie przewidzieć wyniku skonstruowanego przez nas polecenia możemy użyć parametru -WhatIf, który pozwala sprawdzić co zrobi dane polecenie.

Co robić gdy nie wiesz co robić

Jeżeli nie wiesz lub nie pamiętasz jak brzmiało polecenie, którego właśnie potrzebujesz nie potrzebujesz korzystać z dokumentacji. Z pomocą przychodzi cmdlet Get-Command.

Przykład:

PS> Get-Command *process*

Co robić gdy nie wiesz co robić

Kolejnym przydanym poleceniem jest cmdlet Get-Help, zwraca on informacje o działaniu innego polecenia.

W związku z obiektową naturą PowerShell pomocne może okazać się polecenie Get-Member. Zwraca ono informacje o właściwościach I metodach obiektów

Język I środowisko PowerShell

PowerShell obsługuje standardowe operatory arytmetyczne

Komentowanie lini odbywa się tak jak w skryptach bash za pomocą znaku #

Deklaracja zmiennej $a=”wartosc”

Rzutowanie [int] (3/2) da wynik 2. Powershell zaokrągla.

Język I środowisko PowerShell

Zmienne logiczne:

$true – prawda
$false – fałsz
$null – fałsz
Liczba niezerowa – prawda
0 - fałsz

Język I środowisko PowerShell

Definiowanie Tablic:

$tablica=@()

$tablica = 1 , 2 , “trzy” , 3.14

Macierz:

$macierz = @ ((1,2,3),(4,5,6))

Język I środowisko PowerShell

Dostęp do Tablic:

$tablica = 1 , 2 , “trzy” , 3.14

$tablica[0] – zerowy element tablicy

$tablica[-1] – zwraca ostatni element tablicy

$tablica[0..2] – zakres tablicy od 0 do 2

Język I środowisko PowerShell

Tablice asocjacyjne

Tablice umożliwiają kojarzenie kluczy z wartościami

$tablica=@{ }

$tablica=@{klucz1 = “wartosc1”; klucz2 = 3.14 }

Instrukcje warunkowe

```
if(warunek)
{
    Blok instrukcji
}
elseif(warunek)
{
    Blok instrukcji
}
else(warunek)
{
    Blok instrukcji
}
```

Instrukcje pętli

Petla for

```
for(inicjalizacja; warunek ; przyost)
    {
        Blok instrukcji
    }
```

Instrukcje break I continue moga okreslac cel w postaci etykiety

Instrukcje pętli

Pętla while

```
while(warunek)
    {
        Blok instrukcji
    }
```

Dopoki warunek jest prawdziwy PowerShell wykonuje instrukcje

Instrukcje pętli

Pętla do while I do until

```
do
    {
        Blok instrukcji
    } while(warunek)

do
    {
        Blok instrukcji
    } until(warunek)
```

Instrukcje pętli

Pętla foreach

```
foreach($element in $tablica)
    {
        $element
    }
```

Metody statyczne

Aby wywołać statyczną metodę klasy :

[NazwaKlasy]::NazwaMetody(lista parametrow)

Np.

[System.DataTime]::Now

Argumenty

Kazdy skrypt PowerShell można wywołać z parametrami.

Argrumenty zapisywane sa w tablicy args

$args[0] – pierwszy argument

$args.Count – liczba argumentow

Funkcje

```
Function Zasieg:nazwa(parametry)
    {
        Blok instrukcji
    }
```

Typy zasiegów :

- Global
- Script
- local

Funkce wywołujemy tak jak osobny skrypt!

Skrypty