Foreach-Object - info

Zobaczysz, jak wywoływać metody obiektów w potoku

Zobaczysz, czym różni się składnia uproszczona ForEach od składni złożonej

Dowiesz się, jak uruchamiać polecenie dla każdego obiektu z potoku

lub uruchamiać je tylko raz przed przetwarzaniem pierwszego obiektu

lub po przetworzeniu ostatniego obiektu

Wiesz, że obiekty zwracane przez cmdlety posiadają właściwości i metody

Pokaże, jak można pracować z tymi obiektami w potoku w szczególności jak wywoływać metody tych obiektów w potoku

Zaczynamy od

```
1   Get-ChildItem C:\Users
```

```
    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        03.07.2024     15:25                admin
d-r---        26.06.2024     10:15                Public
-a----        03.07.2024     17:52           3286 service.csv
-a----        03.07.2024     17:52           3286 service.txt
-a----        03.07.2024     18:03           1642 services-exported.csv
-a----        03.07.2024     17:56            246 srv.csv
```

Wykasujemy zaznaczone pliki

Można to zrobić przez Remove-Item ale wykorzystamy Get-ChildItem

Sprawdźmy czy Get-ChildIte posiada metodę do usunięcia plików z dysku

```
2   Get-ChildItem C:\Users | Get-Member
```

```
BaseName                   ScriptProperty System.Object BaseN


   TypeName: System.IO.FileInfo

Name                       MemberType     Definition
----                       ----------     ----------
LinkType                   CodeProperty   System.String LinkT
Mode                       CodeProperty   System.String Mode{
Target                     CodeProperty   System.Collections.
AppendText                 Method         System.IO.StreamWri
CopyTo                     Method         System.IO.FileInfo
Create                     Method         System.IO.FileStrea
CreateObjRef               Method         System.Runtime.Remo
CreateText                 Method         System.IO.StreamWri
Decrypt                    Method         void Decrypt()
Delete                     Method         void Delete()
Encrypt                    Method         void Encrypt()
```

Usunięcie z użyciem prostszej składni

```
3   Get-ChildItem C:\Users -Filter "*.csv" | ForEach Delete
4   Get-ChildItem C:\Users
```

```
PS C:\Windows\system32> Get-ChildItem C:\Users -Filter "*.csv" | ForEach Delete

PS C:\Windows\system32> Get-ChildItem C:\Users


    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        03.07.2024     15:25                admin
d-r---        26.06.2024     10:15                Public
-a----        03.07.2024     17:52           3286 service.txt
```

Polecenie nie daje rozbudowanych możliwości

Z użyciem rozbudowanej składni

W pierwszym kroku przeniesiemy pliki tekstowe z C:\Users do C:

```
5   Get-ChildItem C:\Users -Filter "*.txt" | ForEach { $PSItem.CopyTo('c:'+$PSItem.Name) }
```

```
PS C:\Windows\system32> Get-ChildItem C:\Users -Filter "*.txt" | ForEach { $PSItem.CopyTo('c:'+$PSItem.Name) }

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        03.07.2024     17:52           3286 service.txt
```

Budujemy skrypt uruchamiany co godzinę który ma do pliku C:\Users\raport.txt o procesach uruchomionych w danej chwili i ilości wykorzystywanej przez nie pamięci, plik nie ma być nadpisywany a kolejne porcje informacji przy każdym uruchomieniami mają być dopisywane na koniec pliku, aby zapisy się nie wymieszały na początku skryptu przed informacjami o procesach ma znaleźć się data

```
6   Get-Process | ForEach -Begin { Get-Date | Out-File -FilePath 'C:\Users\raport.txt' -Append} -Process { $PSItem | Select-Object Name,VM | Out-File 'C:\Users\raport.txt' -Append }
```

```
PS C:\WINDOWS\system32> Get-Process | ForEach -Begin { Get-Date | Out-File -FilePath 'C:\Users\raport.txt' -Append} -Process { $PSItem | Select-Object Name,VM | Out-File 'C:\Users\raport.t
```

Get-Process | ForEach -Begin { Get-Date | Out-File -FilePath 'C:\Users\raport.txt' -Append} -Process { $PSItem | Select-Object Name,VM | Out-File 'C:\Users\raport.txt' -Append }

Obejrzymy plik

```
7   Get-Content C:\Users\raport.txt
```

```
PS C:\Windows\system32> Get-Content C:\Users\raport.txt
środa, 3 lipca 2024 19:02:51


Name                         VM
----                         --
AggregatorHost 2203387199488


Name                            VM
----                            --
backgroundTaskHost 2203484835840


Name              VM
----              --
csrss 2203412725760
```

Na początku data a potem informacje o procesach

Przeanalizujmy polecenie

Get-Process

| ForEach - dla każdego

-Begin – tylko jeden raz

{ Get-Date | Out-File -FilePath 'C:\Users\raport.txt' bieżącą datę do pliku wysyła

-Append} - dopisując, jeśli ten plik już istnieje

-Process -dla każdego obiektu

{ $PSItem - obiekt procesu

| Select-Object Name,VM wybieramy właściwości

| Out-File 'C:\Users\raport.txt' -Append }

Przykłady zastosowań:

dla każdej grupy wyświetl członków

dla każdego dysku sprawdź obecność pewnych plików

itd. …

Zobaczyłeś, jak wywoływać metody na rzecz obiektów

Czym różni się składnia uproszczona ForEach od składni złożonej

Dowiedziałeś się, jak uruchamiać polecenie dla każdego obiektu z potoku

lub uruchamiać je tylko raz przed przetwarzaniem pierwszego obiektu

lub po przetworzeniu ostatniego obiektu

Użyte polecenia

Get-ChildItem C:\Users

Get-ChildItem C:\Users | Get-Member

Get-ChildItem C:\Users -Filter "*.csv" | ForEach Delete

Get-ChildItem C:\Users

Get-ChildItem C:\Users -Filter "*.txt" | ForEach { $PSItem.CopyTo('c:'+$PSItem.Name) }

Get-Process | ForEach -Begin { Get-Date | Out-File -FilePath 'C:\Users\raport.txt' -Append} -Process { $PSItem | Select-Object Name,VM | Out-File 'C:\Users\raport.txt' -Append }

Get-Content C:\Users\raport.txt