Where-Object - info

Dowiesz się, jak budować wrażenia logiczne

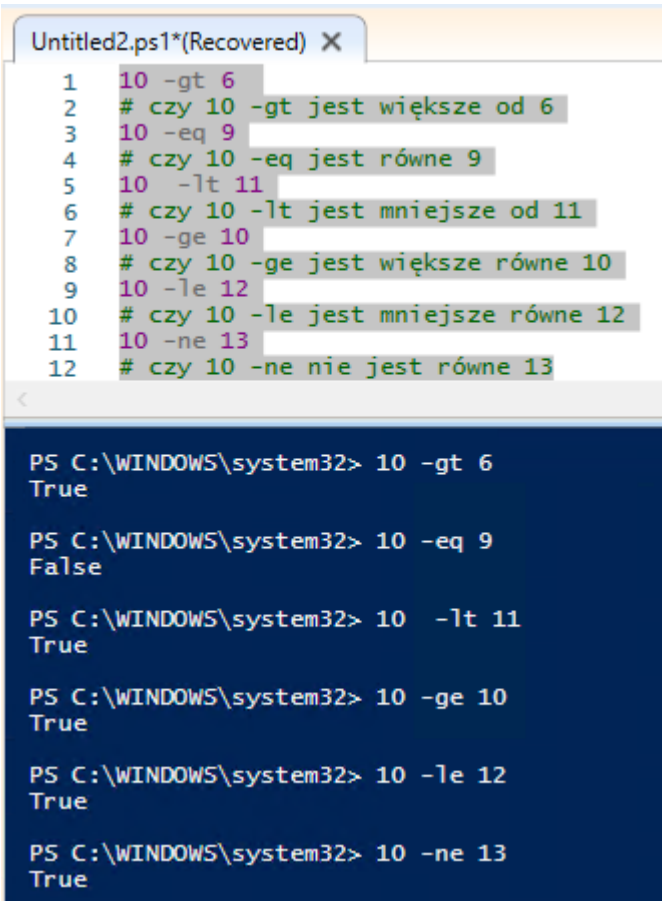
Poznasz polecenie Where-Object

Zobaczysz różnice w składni uproszczonej i zaawansowanej dla Where

Dowiesz się, jak konstruować potoki, by działały wydajnie

W tej lekcji zajmiemy się filtrowaniem wyników z potoku co wymaga budowania wyrażeń logicznych

W PowerShell nie można używać do budowania wyrażeń logicznych symboli matematycznych jak = > < itd. Natomiast doświadczenie ze skryptowaniem w systemach unix w tym Linux są na miejscu



Do porównywania zmiennych znakowych można stosować polecenia powyżej

Zmienne napisowe można porównywać z maskami budowanymi za pomocą metaznaków

Poniżej sprawdzamy czy 'powershell' pasuje do maski 'Power*' używając operatora like

```
13  'powershell' -like 'Power*'
```

```
PS C:\WINDOWS\system32> 'powershell' -like 'Power*'
True
```

Poniżej sprawdzamy czy 'powershell' pasuje do maski 'Power*' używając operatora clike - oznacza rozróżnianie małych i dużych liter

```
14  'powershell' -clike 'Power*'
```

```
PS C:\Windows\system32> 'powershell' -clike 'Power*'
False
```

Kolejny operator -not

-not $true - nieprawda, że prawda

```
15  -not $true
```

```
PS C:\WINDOWS\system32> -not $true
False
```

1 -eq 1 – czy 1 = 1

-and – czy jednocześnie zachodzi warunek po lewej i po prawej

2 -eq 2 czy 2 = 2

```
16  1 -eq 1 -and 2 -eq 2
```

```
PS C:\WINDOWS\system32> 1 -eq 1 -and 2 -eq 2
True
```

```
17  1 -eq 1 -or 2 -eq 20
```

```
PS C:\WINDOWS\system32> 1 -eq 1 -or 2 -eq 20
True
```

Poniżej kolejność działań, najpierw w nawiasach

```
18  -not (1 -eq 1)
```

```
PS C:\WINDOWS\system32> -not (1 -eq 1)
False
```

```
19  -not (1 -eq 1) -and 2 -eq 2
```

```
PS C:\WINDOWS\system32> -not (1 -eq 1) -and 2 -eq 2
False
```

Teraz rzeczywiste obiekty w PowerShell

Wyświetlimy katalogi podczas listowania folderu nie korzystając z filtrowania Get-ChildItem

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

   TypeName: System.IO.DirectoryInfo

Name                      MemberType     Definition
----                      ----------     ----------
LinkType                  CodeProperty   System.String LinkType{g
Mode                      CodeProperty   System.String Mode{get=M
Target                    CodeProperty   System.Collections.Gener
Create                    Method         void Create(), void Crea
CreateObjRef              Method         System.Runtime.Remoting.
CreateSubdirectory        Method         System.IO.DirectoryInfo
Delete                    Method         void Delete(), void Dele
EnumerateDirectories      Method         System.Collections.Gener
EnumerateFiles            Method         System.Collections.Gener
EnumerateFileSystemInfos  Method         System.Collections.Gener
Equals                    Method         bool Equals(System.Objec
GetAccessControl          Method         System.Security.AccessCo
GetDirectories            Method         System.IO.DirectoryInfo[
GetFiles                  Method         System.IO.FileInfo[] Get
GetFileSystemInfos        Method         System.IO.FileSystemInfo
GetHashCode               Method         int GetHashCode()
GetLifetimeService        Method         System.Object GetLifetim
GetObjectData             Method         void GetObjectData(Syste
GetType                   Method         type GetType()
InitializeLifetimeService Method         System.Object Initialize
MoveTo                    Method         void MoveTo(string destD
Refresh                   Method         void Refresh()
SetAccessControl          Method         void SetAccessControl(Sy
ToString                  Method         string ToString()
PSChildName               NoteProperty   string PSChildName=admin
PSDrive                   NoteProperty   PSDriveInfo PSDrive=C
PSIsContainer             NoteProperty   bool PSIsContainer=True
PSParentPath              NoteProperty   string PSParentPath=Micr
```

3

Jak widać powyżej katalogi TypeName: System.IO.DirectoryInfo mają właściwość PSIsContainer=True

Pliki ja widać poniżej dla TypeName: System.IO.FileInfo mają właściwość PSIsContainer=False

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

OpenRead                  Method         System.IO.FileStream OpenRead()
OpenText                  Method         System.IO.StreamReader OpenText()
OpenWrite                 Method         System.IO.FileStream OpenWrite()
Refresh                   Method         void Refresh()
Replace                   Method         System.IO.FileInfo Replace(string dest
SetAccessControl          Method         void SetAccessControl(System.Security.
ToString                  Method         string ToString()
PSChildName               NoteProperty   string PSChildName=service.csv
PSDrive                   NoteProperty   PSDriveInfo PSDrive=C
PSIsContainer             NoteProperty   bool PSIsContainer=False
PSParentPath              NoteProperty   string PSParentPath=Microsoft.PowerShe
```

Ta właściwość rozróżnia plik od katalogu

Komenda poniżej

Get-ChildItem C:\Users - Wyświetla zawartość C:\Users

| - przesyła do komendy poniżej

Where-Object – sprawdza czy -Property PSIsContainer jest -eq równa -Value wartości $true prawda

Jeśli tak to rekord jest wyświetlany na ekranie, a jeśli nie to dane giną i nie są nigdzie przesyłane]

Wynik - tylko wyświetlone foldery:

```
21  Get-ChildItem C:\Users | Where-Object -Property PSIsContainer -EQ -Value $true
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         03.07.2024     15:25                admin
d-r---         26.06.2024     10:15                Public
```

Skrócone polecenie

```
22  Get-ChildItem C:\Users | Where PSIsContainer -EQ $true
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         03.07.2024     15:25                admin
d-r---         26.06.2024     10:15                Public
```

PSIsContainer jest wartoscią logiczną więc nie ma konieczności porównywania z $true lub $false

```
23  Get-ChildItem C:\Users | Where PSIsContainer
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         03.07.2024     15:25                admin
d-r---         26.06.2024     10:15                Public
```

Alias do Where to ?

```
24  Get-ChildItem C:\Users | ? PSIsContainer
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         03.07.2024     15:25                admin
d-r---         26.06.2024     10:15                Public
```

Ta notacja nie radzi sobie z zawansowanymi warunkami

```
25  Get-ChildItem C:\Users | Where PSIsContainer -eq $false -and extension -eq '.txt'
```

```
PS C:\WINDOWS\system32> Get-ChildItem C:\Users | Where PSIsContainer -eq $false -and extension -
Where-Object : Cannot bind parameter because parameter 'eq' is specified more than once. To prov
s that can accept multiple values, use the array syntax. For example, "-parameter value1,value2,
At line:1 char:72
+ ... m C:\Users | Where PSIsContainer -eq $false -and extension -eq '.txt'
+                                                                ~~~
    + CategoryInfo          : InvalidArgument: (:) [Where-Object], ParameterBindingException
    + FullyQualifiedErrorId : ParameterAlreadyBound,Microsoft.PowerShell.Commands.WhereObjectCom
```

W notacji zawansowanej dla polecenia Where należy podać definicję filtru, który będzie szacowany dla każdego przetwarzanego obiektu.

Filtr definiujesz przekazując parametr -FilterScript tu w {} definiujesz wyrażenia, które ma zwracać true lub false

Jeżeli masz się odwołać do aktualnie przetwarzanego w potoku obiektu to należy korzystać z zmiennej $PSItem lub $_

Skrypt będzie wykonywany tyle razy, ile obiektów przechodzi potokiem do Where

Za każdym razem w zmiennej $PSItem lub $_ będzie znajdował się obiekt, który jest przetwarzany

PSItem.PSIsContainer dla każdego przetwarzanego pliku wskazuje na to czy ten obiekt jest plikiem czy folderem

Skrypt przepuści w potoku tylko te obiekty, dla których wyszacuje wartość wyrażenia logicznego na true

Wynik

```
26  Get-ChildItem C:\Users | Where -FilterScript { $PSItem.PSIsContainer -eq $false }
```

```
    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-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.txt
```

Wyświetlone tylko pliki

$_ - też działa - poniżej

```
27  Get-ChildItem C:\Users | Where -FilterScript { $_.PSIsContainer -eq $false }
```

```
    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-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.txt
```

Wróćmy do niedokończonego przykładu

```
28  Get-ChildItem C:\Users | Where -FilterScript { $_.PSIsContainer -eq $false -and $_.Extension -eq '.txt' }
```

```
    Directory: C:\Users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         03.07.2024     17:52           3286 service.txt
-a----         03.07.2024     17:56            246 srv.txt
```

Wyświetlone tylko pliki *.txt

Uwaga wydajnościowa

Filtrowanie w potokach należy wykonywać jak najwcześniej dzięki temu kolejne polecenia nie muszą przetwarzać obiektów które muszą zostać pominięte

Modyfikacja ze względu na wydajność

```
28  Get-ChildItem C:\Users | Where -FilterScript { $_.PSIsContainer -eq $false -and $_.Extension -eq '.txt' }
29  Get-ChildItem C:\Users -Filter "*.txt" -File
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        03.07.2024     17:52           3286 service.txt
-a----        03.07.2024     17:56            246 srv.txt
```

Wynik ten sam, ale gdyby do przetworzenia było tysiące plików to ta druga forma będzie wydajniejsza

Dowiedziałeś się, jak budować wyrażenia logiczne

Poznałeś polecenie Where-Object

Zobaczyłeś różnice w składni uproszczonej i zaawansowanej dla Where

Dowiedziałeś się, jak konstruować potoki, by działały wydajnie

Użyte polecenia

10 -gt 6

czy 10 -gt jest większe od 6

10 -eq 9

czy 10 -eq jest równe 9

10 -lt 11

czy 10 -lt jest mniejsze od 11

10 -ge 10

czy 10 -ge jest większe równe 10

10 -le 12

czy 10 -le jest mniejsze równe 12

10 -ne 13

czy 10 -ne nie jest równe 13

'powershell' -like 'Power*'

'powershell' -like 'Power*'

-not $true

```
1 -eq 1 -and 2 -eq 2

1 -eq 1 -or 2 -eq 20

-not (1 -eq 1)

-not (1 -eq 1) -and 2 -eq 2

Get-ChildItem C:\Users | Get-Member

Get-ChildItem C:\Users | Where-Object -Property PSIsContainer -eq -Value $true

Get-ChildItem C:\Users | Where PSIsContainer -eq $true

Get-ChildItem C:\Users | Where PSIsContainer

Get-ChildItem C:\Users | ? PSIsContainer

Get-ChildItem C:\Users | Where PSIsContainer -eq $false -and extension -eq '.txt'

Get-ChildItem C:\Users | Where -FilterScript { $PSItem.PSIsContainer -eq $false }

Get-ChildItem C:\Users | Where -FilterScript { $_.PSIsContainer -eq $false }

Get-ChildItem C:\Users | Where -FilterScript { $_.PSIsContainer -eq $false -and $_.Extension -eq '.txt'
}

Get-ChildItem C:\Users -Filter "*.txt" -File
```