Select-Object

Modifier les résultats

Il est possible d'utiliser *select-object* pour manipuler certain des champs que l'on veut présenter. Dans cette exemple, après que nous allons lire ce qu'il se trouve dans le fichier ip.txt, qui contiendra une liste d'adresse IP. Nous allons récupérer ce résultat via un pipeline et retournée une valeur boolean qui proviendra d'une réponse du commandlet test-connection.

```
Get-Content .\ip.txt | Select-Object  @{n="Server
IP";e={$_}},@{n="ping?";e={[bool](Test-Connection $_ -Quiet -Count 1)}}
```

Dans cette exemple, j'ai récupéré cette fonction (ici) pour pouvoir transformer le résultat de la valeur lenght renvoyé lors d'un ***get-childitem***, on peut voir aussi qu'il est possible d'utiliser des variable pour simplifier l'usage de la transformation avec ***select-object***.

```
Function Format-FileSize() {
Param ([int32]$value)
If      ($value -gt 1TB)    {[string]::Format("{0:0.00} TB", $Value / 1TB)}
ElseIf  ($Value -gt 1GB)    {[string]::Format("{0:0.00} GB", $Value / 1GB)}
ElseIf  ($Value -gt 1MB)    {[string]::Format("{0:0.00} MB", $Value / 1MB)}
ElseIf  ($Value -gt 1KB)    {[string]::Format("{0:0.00} kB", $Value / 1KB)}
ElseIf  ($Value -gt 0)      {[string]::Format("{0:0.00} B", $Value)}
Else    {""}
}
# Add a custom property to calculate the size in KiloBytes of each FileInfo
object you pass in.
# Use the pipeline variable to divide each file's length by a function
$size = @{label="Size";expression={Format-FileSize($_.Length)}}
# Create an additional calculated property with the number of Days since the
file was last accessed.
# You can also shorten the key names to be 'l', and 'e', or use Name instead
of Label.
$days = @{l="Days";e={((Get-Date) - $_.LastAccessTime).Days}}
# You can also shorten the name of your label key to 'l' and your expression
key to 'e'.
$UserData = Join-Path -Path $env:HOMEDRIVE -ChildPath $env:HOMEPATH
$Documents = (Get-ChildItem -Path $UserData -Filter '*Doc*' -Directory -
ReadOnly).FullName
Get-ChildItem -path $Documents -file | Select-Object Name, $size, $days
```

Voici un autre exemple avec le Get-PSDRIVE:

```
Function Format-FileSize() {
Param ($value)
If      ($value -gt 1TB)    {[string]::Format("{0:0.00} TB", $Value / 1TB)}
ElseIf  ($Value -gt 1GB)    {[string]::Format("{0:0.00} GB", $Value / 1GB)}
```

```
ElseIf  ($Value -gt 1MB)    {[string]::Format("{0:0.00} MB", $Value / 1MB)}
ElseIf  ($Value -gt 1KB)    {[string]::Format("{0:0.00} kB", $Value / 1KB)}
ElseIf  ($Value -gt 0)      {[string]::Format("{0:0.00} B", $Value)}
Else    {}
}

$letterdrive = @{l="DriveLetter";e={$_.Name + ':\'}}
$FreeSpace = @{l="FreeSpace";e={Format-FileSize($($_.Free))}}
$UsedSpace = @{l="UsedSpace";e={Format-FileSize($($_.Used))}}
$Network = @{l="NetDrive";e={if ($_.DisplayRoot -match
'\\'){$true}else{$false}}}
$TotalSpace = @{l="TotalSpace";e={Format-FileSize(($_.Used)+($_.Free))}}
$NetworkPath = @{l="NetworkPath";e={if ($_.DisplayRoot -match
'\\'){$_.displayroot}else{$null}}}
Get-PSDrive | Where-Object { $_.Free -gt 1 } | Select-Object
$letterdrive,$FreeSpace,$UsedSpace,$TotalSpace,$Network,$NetworkPath
```