Instrukcja IF – info

Jak warunkowo wykonywać fragment skryptu

Jak korzystać 1 IF. ELSE. ELSEIF

Kiedy warto korzystać z Instrukcji warunkowe IF

Przyjrzymy się poleceniu warunkowemu które pozwoli zmienić sterowanie w skrypcie w zależności od aktualnej wartości zmiennej

Zanim to polecenie to przypomnimy kilka informacji o Get-WmiObject

```
2   Get-WmiObject Win32_logicaldisk
```

```
PS C:\WINDOWS\system32> $leter='c:'

PS C:\WINDOWS\system32> Get-WmiObject Win32_logicaldisk


DeviceID     : A:
DriveType    : 2
ProviderName :
FreeSpace    :
Size         :
VolumeName   :

DeviceID     : C:
DriveType    : 3
ProviderName :
FreeSpace    : 1569577967616
Size         : 1590611619840
VolumeName   :

DeviceID     : D:
DriveType    : 5
ProviderName :
FreeSpace    :
Size         :
VolumeName   :
```

Zwracane są informacje o instancjach dysku logicznego na tym komputerze

-Filter ogranicza ilość zwracanych wyników wskazując na jeden napęd dyskowy

```
3   Get-WmiObject Win32_logicaldisk -Filter "DeviceID='c:'"
4
```

```
PS C:\WINDOWS\system32> Get-WmiObject Win32_logicaldisk -Filter "DeviceID='c:'"


DeviceID     : C:
DriveType    : 3
ProviderName :
FreeSpace    : 1569576984576
Size         : 1590611619840
VolumeName   :
```

To samo, ale z użycie zmiennej $leter

```
4   Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'"
```

```
PS C:\WINDOWS\system32> Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'"


DeviceID     : C:
DriveType    : 3
ProviderName :
FreeSpace    : 1569576726528
Size         : 1590611619840
VolumeName   :
```

Wynik ten sam co wyżej, czyli polecenie działa.

Jeżeli przyjrzymy się co jest zwracane w intencji dysku logicznego to znajdziemy tu

```
5   Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | Get-Member
```

```
   TypeName: System.Management.ManagementObject#root\cimv2\Win32_LogicalDisk

Name                       MemberType     Definition
----                       ----------     ----------
PSComputerName             AliasProperty  PSComputerName = __SERVER
Chkdsk                     Method         System.Management.ManagementBaseObject Chkdsk(System.Boole...
Reset                      Method         System.Management.ManagementBaseObject Reset()
SetPowerState              Method         System.Management.ManagementBaseObject SetPowerState(Syste...
Access                     Property       uint16 Access {get;set;}
Availability               Property       uint16 Availability {get;set;}
BlockSize                  Property       uint64 BlockSize {get;set;}
Caption                    Property       string Caption {get;set;}
Compressed                 Property       bool Compressed {get;set;}
ConfigManagerErrorCode     Property       uint32 ConfigManagerErrorCode {get;set;}
ConfigManagerUserConfig    Property       bool ConfigManagerUserConfig {get;set;}
CreationClassName          Property       string CreationClassName {get;set;}
Description                Property       string Description {get;set;}
DeviceID                   Property       string DeviceID {get;set;}
DriveType                  Property       uint32 DriveType {get;set;}
ErrorCleared               Property       bool ErrorCleared {get;set;}
ErrorDescription           Property       string ErrorDescription {get;set;}
ErrorMethodology           Property       string ErrorMethodology {get;set;}
FileSystem                 Property       string FileSystem {get;set;}
FreeSpace                  Property       uint64 FreeSpace {get;set;}
InstallDate                Property       string InstallDate {get;set;}
LastErrorCode              Property       uint32 LastErrorCode {get;set;}
MaximumComponentLength     Property       uint32 MaximumComponentLength {get;set;}
MediaType                  Property       uint32 MediaType {get;set;}
Name                       Property       string Name {get;set;}
QuotasRebuilding           Property       bool QuotasRebuilding {get;set;}
Size                       Property       uint64 Size {get;set;}
Status                     Property       string Status {get;set;}
```

Znajdziemy tu właściwości

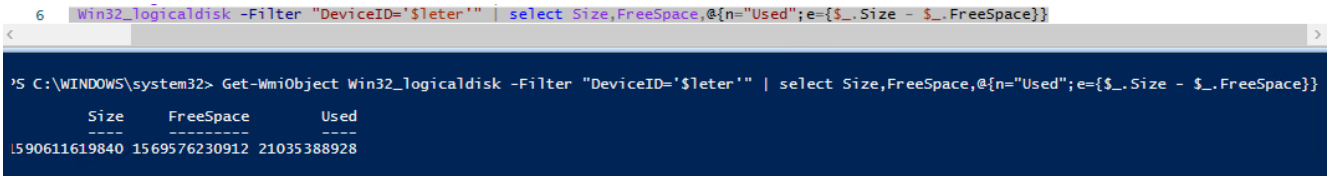
FreeSpace – ilość wolnego miejsca na dysku

Size - ilość miejsca na dysku

Odejmowanie Size – FreeSpace = ilość miejsca zajętego

Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select Size,FreeSpace,@{n="Used";e={$_.Size - $_.FreeSpace}}

Wszystkie wartości są podawane w bajtach



Jesteśmy na etapie pisania skryptu, w którym przez parametr będzie można przekazywać nie tylko literę dysku, ale również co ma być zwrócone.

Jeśli przekażesz napis "Used" to zwracać będziemy informacje o ilości używanego miejsca na dysku.

Jeśli przekażesz tekst "FreeSpace" to znaczy, że interesuje nas ilość wolnego miejsca na dysku.

Jeśli przekażesz tekst "TotalSpace" to znaczy, że interesuje nas całkowita ilość miejsca na dysku.

Posłużymy się w naszym skrypcie instrukcją warunkową if która badać będzie co znajduje się w zmiennej $valueToReturn

Jeśli w tej zmiennej znajduje się napis "FreeSpace" to wtedy z obiektu logicaldisk dla danego dysku wybierzemy tylko właściwość "FreeSpace"

W przeciwnym wypadku, jeżeli w zmiennej będzie napis "TotalSpace" to wtedy z obiektu logicaldisk dla danego dysku wybierzemy tylko właściwość "Size"

W przeciwnym wypadku, a więc nie tylko w przypadku, gdy ktoś przesyła tekst "Used" ale w każdym pozostałym przypadku, gdy w parametrze będzie inny napis niż "FreeSpace" lub napis "TotalSpace" to skrypt dla danego dysku zwróci ilość miejsca zajętego na dysku.

```
1  $leter='c:'
2  $valueToReturn="Used"
3  If($valueToReturn -eq "FreeSpace")
4  {
5  Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select FreeSpace
6  }
7  ElseIf($valueToReturn -eq "TotalSpace")
8  {
9  Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select Size
10 }
11 Else
12 {
13 Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select Size,FreeSpace,@{n="Used";e={$_.Size - $_.FreeSpace}}
14 }
```

Widać, że instrukcja składa się każdorazowo z wyrażenia if po którym w nawiasie umieszczamy wyrażenie, które po zinterpretowaniu ma przyjąć wartość prawda lub fałsz.

Jeśli wyliczona wartość to prawda to wykona się blok kodu pod instrukcją If.

Jeśli wyliczona wartość to fałsz skrypt opuści ten blok i przeniesie się do kolejnych instrukcji którymi mogą być ElseIf, Else a w przypadku ich braku kolejne instrukcje skryptu

Polecenie If a dokładnie wyrażenie warunkowe używane w If decyduje który fragment skryptu zostanie w danej sytuacji wykonany a który nie.

Oto wynik:

```powershell
$leter='c:'
$valueToReturn="Used"

If($valueToReturn -eq "FreeSpace")
{
Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select FreeSpace
}
ElseIf($valueToReturn -eq "TotalSpace")
{
Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select Size
}
Else
{
Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select Size,FreeSpace,@{n="Used";e={$_.Size - $_.FreeSpace}}
}
```

```
{
Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select FreeSpace
}
ElseIf($valueToReturn -eq "TotalSpace")
{
Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select Size
}
Else
{
Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select Size,FreeSpace,@{n="Used";e={$_.Size - $_.FreeSpace}}
}


         Size     FreeSpace        Used
         ----     ---------        ----
1590611619840 1569570381824 21041238016
```

Dokładniej:

```
         Size     FreeSpace        Used
         ----     ---------        ----
1590611619840 1569570381824 21041238016
```

Skrypt zwrócił, ile miejsca jest zajętego, a więc informacje "Used"

Jeżeli interesuje cię, ile miejsca na dysku jest wolnego to zmień wartość parametru $valueToReturn="FreeSpace" i uruchom skrypt

Wynik

```
     FreeSpace
     ---------
156957415833 6
```

- ilość wolnego miejsca na dysku.

Jeżeli interesuje cię, całkowita wielkość dysku to zmień wartość parametru $valueToReturn="TotalSpace" i uruchom skrypt

Wynik

```
         Size
         ----
1590611619840
```

- całkowita wielkość dysku.

Polecenie IF nadaje się do funkcji których działanie zależy od wartość przekazywanych parametrów.

W przykładzie dokonaliśmy uproszczenia, decydujące znaczenie zamiast parametrów miały lokalnie zdefiniowane zmienne, zasada działania polecenia if pozostaje taka sama.

Używaj If jeżeli w skrypcie chcesz zareagować na zmieniające się warunki w których będzie pracował skrypt, lepiej napisać jeden skrypt który dzięki If będzie obsługiwał więcej środowisk, serwerów niż utrzymywać kilka bardzo podobnych skryptów zaprojektowanych indywidualnie dla każdego środowiska czy serwera, np. badaj czy istnieje plik i reaguj w zależności od tego czy on jest czy go nie ma, sprawdzaj ilość wolnej pamięci w systemie i obsługuj zdarzenia inaczej gdy masz dużo pamięci a inaczej kiedy masz mało pamięci, sprawdź czy użytkownik jest lokalnym administratorem i zareaguj na to w zależności od sytuacji.

Zobaczyłeś jak warunkowo wykonywać fragment skryptu

Wiesz, jak korzystać: IF. ELSE. ELSEIF

Usłyszałeś, kiedy warto korzystać z instrukcji warunkowej IF

Użyte polecenia

```
$leter='c:'

Get-WmiObject Win32_logicaldisk

Get-WmiObject Win32_logicaldisk -Filter "DeviceID='c:'"

Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'"

Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | Get-Member

Get-WmiObject Win32_logicaldisk -Filter "DeviceID='$leter'" | select
Size,FreeSpace,@{n="Used";e={$_.Size - $_.FreeSpace}}
```