Pętla FOR – info

Zobaczysz, jak działa pętla for

Co to jest iteracja i zmienna sterująca

Jaka jest różnica między for a foreach

Porównanie pętli foreach z cmd-letem foreach-object

Instrukcja for ma za zadanie wykonywać polecenia umieszczone w nawiasie klamrowym przy każdej iteracji, która zależy najczęściej od pewnego wyrażenia, dobrze nadaje się do wykorzystania, jeżeli pewną czynność masz wykonać określoną ilość razy np. skopiować plik 10 razy, sprawdzić 60 razy na minutę łączność z serwerem, wysłać 5 emaili itp.,

Każdorazowe wykonanie czynności to tzw iteracja, rządzi nią pewien warunek logiczny działający w oparciu o zmienną zwaną zmienną sterującą.

Zmienna sterująca ma ustaloną pewną wartość na początku a następnie z każdą iteracją jest powiększana lub pomniejszana. Zazwyczaj zmienną powiększasz lub pomniejszasz o jeden, ale nie jest to reguła.

Po iteracji sprawdza się czy warunek logiczny jest poprawny i z pętli się wychodzi wtedy, gdy warunek nie jest prawdziwy.

For nie różni się znacznie od instrukcji for w innych językach programowania, może tylko troszeczkę pod względem wykonywania porównań.

W tym przypadku pętla ma się wykonać dla zmiennej sterującej $i na początku równej 0 w taki sposób, że przy każdej iteracji wartość i będzie zwiększana o 1 tak długo aż i nie przekroczy wartości 10. Specyficzny warunek porównania widoczny tutaj mówi, że $i ma być lt czyli mniejszy niż 10. Kiedy $i będzie już większe lub równe 10 pętla zakończy się

```
1  for($i=0;$i -lt 10;$i++)
2  {
3      Write-Host "$i"
4  }
```

```
0
1
2
3
4
5
6
7
8
9
```

Jak widać powyżej zostały wyświetlone liczby od 0 do 9, kiedy i przyjmie wartość 10 warunek $i -lt 10 nie będzie spełniony i pętla przerwie się.

Często for wykorzystywany jest do wykonywania czynności dla każdego elementu tablicy.

Zmienna serująca wskazuje wtedy na aktualnie przetwarzany element z tablicy.

Kiedy zmienna sterująca przyjmuje wartość większą niż ilość elementów z tablicy to pętla kończy się, w takim przypadku podobnie jak w innych językach programowania można skorzystać z polecenia foreach. W tym przypadku najpierw zdefiniowałem listę pięciu napisów a następnie dla każdego elementu z tej listy wyświetlamy napis "Aktualnym elementem jest $x" i tu za xx podstawiamy wartość z listy.

Notacja ta zwalnia z myślenia o elementach sterujących.

Polecenie jest wykonywane dla każdego elementu z listy.

$x w każdej iteracji przyjmuje wartość odpowiadającą aktualnie przetwarzanemu obiektowi z listy.

Nim takie polecenia były dostępne programiści często zapominali o powiększaniu zmiennej sterującej bądź popełniali błędy konstruując warunek zakończenia pętli, tu nie ma tego problemu.

```
6   $list="jeden", "dwa", "trzy", "cztery", "pieć"
7
8   foreach($x in $list)
9   {
10   Write-Host "Aktualnym elementem jest $x"
11  }
```

```
Aktualnym elementem jest jeden
Aktualnym elementem jest dwa
Aktualnym elementem jest trzy
Aktualnym elementem jest cztery
Aktualnym elementem jest pieć
Aktualnym elementem jest jeden
Aktualnym elementem jest dwa
Aktualnym elementem jest trzy
Aktualnym elementem jest cztery
Aktualnym elementem jest pieć
```

Można stosować następującej scenariusze stosowanie polecenia foreach np.:

skopiowanie pliku na każdy serwer z pewnej listy serwerów,

musisz uruchomić każdą usługę z pewnej listy usług,

musisz zaraportować wolne miejsce na dysku dla każdego dysku z listy,

musisz wykonać kopię każdej bazy znajdującej się na serwerze sql,

i wiele innych.

Gdzie tylko to możliwe używaj foreach zamiast for

Podobny efekt można uzyskać stosując cmdlet ForEach-Object, jednak to nie ta sama składnia jak kod implementujący pętlę foreach.

Wartość listy zostanie potokiem przekazana do cemdletu foreach które dla każdej otrzymanej wartości wyświetla "Aktualnym elementem jest i tu wartość tego elementu"

Wyniki są do siebie zbliżone to jednak w przypadku pierwszego polecenia mamy do czynienia z wyrażeniem iterującym a w drugim przypadku z cmdletem foreach, który nie wymaga definiowania zmiennej do określania który element jest w tej chwili przetwarzany, informacja ta jest w zmiennej $_ lub $PSItem

```
8  foreach($x in $list)
9  {
10  Write-Host "Aktualnym elementem jest $x"
11 }
12 $list | foreach {Write-Host "Aktualnym elementem jest $_" }
```

```
{
 Write-Host "Aktualnym elementem jest $x"
}
Aktualnym elementem jest jeden
Aktualnym elementem jest dwa
Aktualnym elementem jest trzy
Aktualnym elementem jest cztery
Aktualnym elementem jest pieć

PS C:\WINDOWS\system32> $list | foreach {Write-Host "Aktualnym elementem jest $_" }
Aktualnym elementem jest jeden
Aktualnym elementem jest dwa
Aktualnym elementem jest trzy
Aktualnym elementem jest cztery
Aktualnym elementem jest pieć
```

Zobaczyłeś, jak działa pętla for

Wiesz co to jest iteracja i zmienna sterująca

Znasz różnicę między for a foreach

Zobaczyłeś porównanie pętli foreach z cmd-Ietem foreach-object