W03 Pipline

Tworzenie funkcji zaawansowanej akceptującej dane z potoku (pipeline)

W PowerShell funkcje zaawansowane mogą przyjmować dane z potoku (pipeline), co umożliwia ich użycie w łańcuchach poleceń. Aby funkcja obsługiwała dane z potoku, należy odpowiednio oznaczyć parametr oraz wykorzystać odpowiednie bloki: Begin, Process, End.

Funkcja zaawansowana zawiera atrybut [CmdletBinding()] oraz może mieć trzy bloki:

Blok	Opis
Begin	Inicjalizacja – wykonywana raz przed przetwarzaniem danych.
Process	Wykonywana dla każdego elementu przekazanego przez pipeline.
End	Zakończenie – wykonywana raz po przetworzeniu wszystkich danych.

Parametr akceptujący dane z pipeline

Aby parametr mógł przyjmować dane z potoku, należy użyć atrybutu **[Parameter(ValueFromPipeline=$true)]**.

Funkcja przyjmująca dane z pipeline, wypisująca je z dodatkiem informacji:

```
function Show-ItemInfo {
    [CmdletBinding()]
    param (
        [Parameter(Mandatory=$true, ValueFromPipeline=$true)]
        [string]$Name
    )

    begin {
        Write-Verbose "Rozpoczynamy przetwarzanie..."
    }

    process {
        Write-Output "Odebrano z pipeline: $Name"
    }

    end {
        Write-Verbose "Zakończono przetwarzanie."
    }
}

# Przykład użycia:
'Serwer1','Serwer2' | Show-ItemInfo -Verbose
```

Przekazywanie danych przez nazwę właściwości (ValueFromPipelineByPropertyName)

Funkcja PowerShell może również odbierać dane z potoku przez dopasowanie nazw właściwości obiektów do nazw parametrów funkcji. W tym celu należy użyć atrybutu

[Parameter(ValueFromPipelineByPropertyName=$true)]. Ten mechanizm jest szczególnie przydatny, gdy dane są przekazywane w postaci obiektów z właściwościami.

```
function Show-UserInfo {
    [CmdletBinding()]
    param (
        [Parameter(ValueFromPipelineByPropertyName=$true)]
        [string]$Username
    )
    process {
        "User: $Username"
    }
}

# Dane wejściowe jako obiekty
[PSCustomObject]@{ Username = 'admin' }, @{ Username = 'guest' } |
Show-UserInfo
```

Przekazywanie danych przez wartość (ValueFromPipeline)

W przypadku przekazywania danych przez wartość, typ parametru musi pasować do typu danych przekazywanych z potoku. Mechanizm ten działa tylko wtedy, gdy cały obiekt (np. string, int) jest przekazywany do funkcji.

Przykład:

```
function Write-Upper {
    [CmdletBinding()]
    param (
        [Parameter(ValueFromPipeline=$true)]
        [string]$Text
    )
    process {
        $Text.ToUpper()
    }
}

'one', 'two', 'three' | Write-Upper
```

Wskazówki praktyczne i dobre praktyki

- Parametry można oznaczyć jednocześnie ValueFromPipeline i ValueFromPipelineByPropertyName, co zwiększa elastyczność funkcji.
- Warto zawsze uwzględniać blok process, gdy oczekuje się danych z pipeline – to właśnie on działa dla każdego elementu.
- Dobrą praktyką jest używanie [CmdletBinding()], co umożliwia obsługę parametrów takich jak -Verbose, -WhatIf, -Confirm.
- W funkcjach przyjmujących dane z pipeline warto testować działanie z różnymi typami wejścia, aby zapewnić ich odporność.