W01- Funkcje Zaawansowane w PowerShell

Funkcje zaawansowane w PowerShell (ang. Advanced Functions) są mechanizmem pozwalającym pisać własne polecenia, które zachowują się jak natywne cmdlety.

Są szczególnie przydatne w sytuacjach, gdy chcemy tworzyć moduły, automatyzować zadania, budować narzędzia administracyjne i korzystać z pełnych możliwości PowerShell, w tym takich opcji jak wsparcie dla potoku, obsługa -WhatIf i -Confirm, czy zaawansowana walidacja parametrów.

Zalety funkcji zaawansowanych:

- Pozwalają tworzyć narzędzia o funkcjonalności cmdletów.
- Umożliwiają wsparcie dla potoku (pipeline).
- Pozwalają na użycie zaawansowanych parametrów i ich walidacji.
- Obsługują standardowe przełączniki PowerShell, jak -WhatIf i -Confirm.

Podstawowa składnia funkcji zaawansowanej

Funkcja zaawansowana korzysta z dyrektywy [CmdletBinding()] i bloku param(). Pozwala to na użycie wielu dodatkowych opcji PowerShell.

```
function Get-Greeting {
    [CmdletBinding()]
    param(
        [Parameter(Mandatory = $true)]
        [string]$Name
       )
    Write-Output "Hello, $Name!"
}
```

Parametr -WhatIf

Parametr **what-if** to jedna z najważniejszych funkcji ochronnych w PowerShell. Pozwala on użytkownikowi sprawdzić, co zostałoby zrobione przez daną funkcję lub polecenie, bez faktycznego wykonania tej operacji. Dzięki temu można przetestować zachowanie funkcji w bezpieczny sposób – bez usuwania plików, restartowania usług, czy modyfikacji systemu.

Aby funkcja mogła obsługiwać -*WhatIf*, należy:

1. Dodać atrybut [CmdletBinding(SupportsShouldProcess = $true)] do funkcji.
2. Użyć metody $PSCmdlet.ShouldProcess() wewnątrz funkcji.

Metoda **`$PSCmdlet.ShouldProcess(<cel>, <działanie>)`**

zwraca $true tylko wtedy, gdy użytkownik:

nie podał parametru **-WhatIf** – funkcja wykona operację,

podał **-WhatIf** – funkcja NIE wykona operacji, ale wypisze co by zrobiła.

Przykład:

```
if ($PSCmdlet.ShouldProcess("plik.txt", "Usunięcie"))
{
   Remove-Item "plik.txt"
}
```

Z *-WhatIf* zostanie wypisane: *What if: Performing the operation "Usunięcie" on target "plik.txt".**

Prametr -Confirm

Zmienna **$ConfirmPreference**

PowerShell wykorzystuje zmienną automatyczną $ConfirmPreference, aby określić, kiedy użytkownik powinien być zapytany o potwierdzenie operacji.

Dostępne wartości:

- High – użytkownik zostanie zapytany tylko przy operacjach oznaczonych jako High.
- Medium – zapytania przy Medium i High.
- Low – zapytania przy Low, Medium i High.
- None – brak zapytań o potwierdzenie (chyba że wymuszone ręcznie).

Domyślna wartość to "High". Można ją zmienić tymczasowo:

```
$ConfirmPreference = 'Low'
```

Cmdlety PowerShell obsługujące -Confirm

Wiele wbudowanych cmdletów obsługuje parametr -Confirm. Przykłady:

- Remove-Item – usuwa pliki lub foldery.
- Stop-Service – zatrzymuje usługę.
- Restart-Computer – restartuje komputer.

Użycie:

Wymuszenie potwierdzenia nawet jeśli **ConfirmImpact** nie wskazuje na potrzebę:

```
Stop-Service -Name Spooler -Confirm:$true
```

Tworzenie własnej funkcji obsługującej -Confirm

Aby funkcja obsługiwała -Confirm, należy:

- dodać [CmdletBinding(SupportsShouldProcess = $true, ConfirmImpact = 'Medium')],
- użyć metody $PSCmdlet.ShouldProcess() – jak przy -WhatIf.

```
function Stop-ServiceSafe {
    [CmdletBinding(SupportsShouldProcess=$true, ConfirmImpact = 'High')]
        param (
            [Parameter(Mandatory = $true)]
            [string]$ServiceName
        )
        if ($PSCmdlet.ShouldProcess($ServiceName, "Zatrzymanie usługi"))
        {
            Stop-Service -Name $ServiceName -Force
        }
    }

    # Przykład wywołania:
# Stop-ServiceSafe -ServiceName 'Spooler' -Confirm
```

Podsumowanie:

- $PSCmdlet.ShouldProcess() służy do integracji z -WhatIf i -Confirm. ShouldContinue() wymusza interaktywny wybór (nawet mimo -Confirm:$false). ConfirmImpact może mieć wartości: Low, Medium, High – wpływa kiedy pojawi się -Confirm.

Write-Host , Write-Output

Write-Host służy do wypisywania informacji bezpośrednio do konsoli użytkownika. Nie przekazuje danych dalej ani nie umożliwia ich zapisania do zmiennej.

Write-Output przesyła dane do strumienia wyjściowego, co umożliwia przekazanie danych do zmiennej, potoku, logowania itp.

```
function Demo-WriteOutput {
    Write-Output "To jest komunikat z Write-Output"
}
```

Wywołanie:

```
$wynik = Demo-WriteOutput
# $wynik zawiera tekst
```

Porównanie działania

```
function Compare-WriteHostOutput {
    Write-Host "Host: To nie trafi do zmiennej"
    Write-Output "Output: To trafi do zmiennej"
}
```

Wywołanie:

```
$result = Compare-WriteHostOutput
$result
# Zawiera tylko dane z Write-Output
```

	Write-Host	Write-Output
Wysyła dane do potoku	❌	✅
Działa z potokiem	❌	✅
Można przypisać do zmiennej	❌	✅
Wyświetla kolorowy tekst	✅	❌
Zalecane do wypisywania wyników	❌	✅
Zalecane do komunikatów dla użyt.	✅	❌