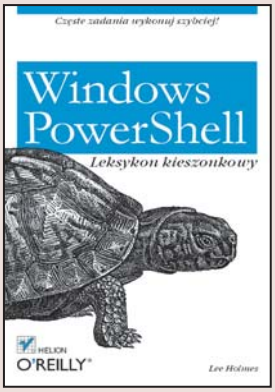
Windows PowerShell. Leksykon kieszonkowy



Autor: Lee Holmes
Tłumaczenie: Grzegorz Werner
ISBN: 978-83-246-2043-2
Tytuł oryginału: Windows PowerShell Pocket Reference: Pocket Reference
Format: 115x170, stron: 168

Częste zadania wykonuj szybciej!

- Jakie polecenia i wyrażenia dostępne są w PowerShell?
- Jak zarządzać błędami?
- Jak wykorzystać klasy .NET oraz .COM?

PowerShell powstał w 2006 roku jako następca takich interpreterów jak command.com czy też cmd.exe, znanych z czasów systemu operacyjnego MS DOS czy też pierwszych wydań Windows XP. Aktualnie dostępny jest dla następujących platform: Windows XP SP2, Windows Vista, Windows 2003. Cechą charakterystyczną PowerShell jest logika obiektowa. Wynikiem każdego polecenia jest obiekt określonego typu, posiadający swoje metody oraz właściwości, które mogą być wykorzystane w dalszym toku przetwarzania.

Dzięki książce „Windows PowerShell. Leksykon kieszonkowy" poznasz język i środowisko PowerShell. Dowiesz się, w jaki sposób wydawać polecenia, jak stosować operatory, instrukcje warunkowe, pętle i wiele elementów znanych z innych języków programowania. Po przeczytaniu tej książki będziesz potrafił zarządzać błędami oraz śledzić i debugować skrypty. Dodatkowo nauczysz się stosować wyrażenia regularne oraz poznasz wybrane klasy .NET oraz .COM. Cała wiedza zostanie przekazana przez jednego z twórców PowerShella, co stanowi gwarancję aktualności, przydatności i poprawności omawianych tematów.

- Polecenia i wyrażenia w powłoce PowerShell
- Wykorzystanie tablic
- Zastosowanie języka XML
- Sposoby uruchamiania skryptów
- Zarządzanie błędami
- śledzenie i debugowanie skryptów
- Zastosowanie wyrażeń regularnych
- Wykorzystanie zmiennych automatycznych w PowerShell
- Klasy .NET i ich zastosowanie
- Klasy .COM i ich zastosowanie
- Sposoby formatowania łańcuchów

Oszczędzaj czas dzięki PowerShell!

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 3. Wyrażenia regularne

Wyrażenia regularne odgrywają ważną rolę w większości zadań wymagających parsowania i dopasowywania tekstu. Stanowią one zasadniczy element operatora `-match`, instrukcji `switch`, cmdletu `Select-String`. W tabelach od 3.1 do 3.9 wymienione są często używane wyrażenia regularne.

Tabela 3.1. Klasy znakowe: wzorce, które reprezentują zbiory znaków

Klasa znakowa	Dopasowuje

`.`	Dowolny znak z wyjątkiem znaku nowego wiersza. Jeśli wyrażenie regularne używa opcji `SingleLine`, dopasowuje dowolny znak.
	`PS >"T" -match '.'`
	`True`

`[`*`znaki`*`]`	Dowolny spośród znaków w nawiasie. Przykład: `[aeiou]`.
	`PS >"Test" -match '[Tes]'`
	`True`

`[^`*`znaki`*`]`	Dowolny znak oprócz tych, które znajdują się w nawiasie. Przykład: `[^aeiou]`.
	`PS >"Test" -match '[^Tes]'`
	`False`

`[`*`początek-koniec`*`]`	Dowolny znak między znakami *początek* i *koniec* (włącznie). W nawiasie można określić wiele zakresów. Na przykład: `[a-eh-j]`.
	`PS >"Test" -match '[e-t]'`
	`True`

`[^`*`początek-koniec`*`]`	Dowolny znak oprócz tych, które znajdują się między znakami *początek* i *koniec* (włącznie). W nawiasie można określić wiele zakresów. Na przykład: `[^a-eh-j]`.
	`PS >"Test" -match '[^e-t]'`
	`False`

Tabela 3.1. Klasy znakowe: wzorce, które reprezentują zbiory znaków (ciąg dalszy)

Klasa znakowa	Dopasowuje

`\p{klasa znakowa}`	Dowolny znak w grupie albo zakresie bloków Unicode określonym przez {*klasę znakową*}.
	`PS >"+" -match '\p{Sm}'`
	`True`

`\P{klasa znakowa}`	Dowolny znak oprócz tych, które należą do grupy albo zakresu bloków Unicode określone przez {*klasę znakową*}.
	`PS >"+" -match '\P{Sm}'`
	`False`

`\w`	Dowolny znak spośród wchodzących w skład słów.
	`PS >"a" -match '\w'`
	`True`

`\W`	Dowolny znak oprócz tych, które wchodzą w skład słów.
	`PS >"!" -match '\w'`
	`True`

`\s`	Dowolny znak odstępu. `PS >"`t" -match '\s'` `True`
`\S`	Dowolny znak oprócz znaków odstępu. `PS >" `t" -match '\S'` `False`

`\d`	Dowolna cyfra dziesiętna.
	`PS >"5" -match '\d'`
	`True`

`\D`	Dowolny znak oprócz cyfr dziesiętnych.
	`PS >"!" -match '\D'`
	`True`

Tabela 3.2. Kwantyfikatory: wyrażenia, które wymuszają liczność poprzedzającego je wyrażenia

Kwantyfikator	Opis

<brak>	Jedno dopasowanie.
	`PS >"T" -match 'T'`
	`True`

*	Zero lub więcej dopasowań. Dopasowywanych jest jak najwięcej elementów.
	`PS >"A" -match 'T*'`
	`True`
	`PS >"TTTTT" -match '^T*$'`
	`True`

+	Jedno lub więcej dopasowań. Dopasowywanych jest jak najwięcej elementów.
	`PS >"A" -match 'T+'`
	`False`
	`PS >"TTTTT" -match '^T+$'`
	`True`

?	Zero lub jedno dopasowanie. Dopasowywanych jest jak najwięcej elementów.
	`PS >"TTTTT" -match '^T?$'`
	`False`

{*n*}	Dokładnie *n* dopasowań.
	`PS >"TTTTT" -match '^T{5}$'`
	`True`

{*n*,}	*n* lub więcej dopasowań. Dopasowywanych jest jak najwięcej elementów.
	`PS >"TTTTT" -match '^T{4,}$'`
	`True`

{*n*,*m*}	Od *n* do *m* dopasowań (włącznie). Dopasowywanych jest jak najwięcej elementów.
	`PS >"TTTTT" -match '^T{4,6}$'`
	`True`

*?	Zero lub więcej dopasowań. Dopasowywanych jest jak najmniej elementów.
	`PS >"A" -match '^AT *?$'`
	`True`

Tabela 3.2. Kwantyfikatory: wyrażenia, które wymuszają liczność poprzedzającego je wyrażenia (ciąg dalszy)

Kwantyfikator	Opis

`+?`	Jedno lub więcej dopasowań. Dopasowywanych jest jak najmniej elementów.
	`PS >"A" -match '^AT +?$'`
	`False`

`??`	Zero lub jedno dopasowanie. Dopasowywanych jest jak najmniej elementów.
	`PS >"A" -match '^AT ??$'`
	`True`

`{n}?`	Dokładnie *n* dopasowań.
	`PS >"TTTTT" -match '^T{5}?$'`
	`True`

`{n,}?`	*n* lub więcej dopasowań. Dopasowywanych jest jak najmniej elementów.
	`PowerShell >"TTTTT" -match '^T{4,}?$'`
	`True`

`{n,m}?`	Od *n* do *m* dopasowań (włącznie). Dopasowywanych jest jak najmniej elementów.
	`PS >"TTTTT" -match '^T{4,6}?$'`
	`True`

Tabela 3.3. Konstrukcje grupujące: wyrażenia, które umożliwiają grupowanie znaków, wzorców i innych wyrażeń

Konstrukcja grupująca	Opis

`(tekst)`	Przechwytuje tekst dopasowany przez wyrażenie w nawiasie. Grupy te są numerowane (zaczynając od jedności) według kolejności nawiasu otwierającego.
	`PS >"Halo" -match '^(.*)lo$';`
	`  $matches[1]`
	`True`
	`Ha`

Tabela 3.3. Konstrukcje grupujące: wyrażenia, które umożliwiają grupowanie znaków, wzorców i innych wyrażeń (ciąg dalszy)

Konstrukcja grupująca	Opis

`(?<nazwa> )`	Przechwytuje tekst dopasowany przez wyrażenie w nawiasie. Grupy te mają nazwy określone przez użytkownika.
	`PS >"Halo" -match '^(?<Jeden>.*)lo$';`
	`    $matches.Jeden`
	`True`
	`Ha`

`(?<nazwa1-` `↪nazwa2> )`	Definicja grupy zrównoważonej. Jest to konstrukcja aawansowana, która umożliwia dopasowywanie zrównoważonych par elementów.

`(?: )`	Grupa nieprzechwytująca.
	`PS >"A1" -match '((A|B)\d)';`
	`    $matches`
	`True`
	`Name              Value`
	`----              -----`
	`2                 A`
	`1                 A1`
	`0                 A1`
	`PS >"A1" -match '((?:A|B)\d)'; $matches`
	`True`
	`Name              Value`
	`----              -----`
	`1                 A1`
	`0                 A1`

`(?imnsx-` `↪imnsx: )`	Włącza lub wyłącza określoną opcję w danej grupie. Oto obsługiwane opcje: `i` dopasowywanie bez uwzględniania wielkości liter `m` dopasowywanie wielowierszowe `n` jawne przechwytywanie `s` dopasowywanie jednowierszowe `x` ignorowanie odstępów `PS >"Te`nst" -match '(T e.st)'` `False` `PS >"Te`nst" -match '(?sx:T e.st)'` `True`

Tabela 3.3. Konstrukcje grupujące: wyrażenia, które umożliwiają grupowanie znaków, wzorców i innych wyrażeń (ciąg dalszy)

Konstrukcja grupująca	Opis

`(?= )`	Pozytywna asercja z patrzeniem w przód. Gwarantuje, że dany wzorzec pasuje do dalszej części tekstu, ale nie przeprowadza rzeczywistego dopasowania.
	`PS >"555-1212" -match '(?=...-)(.*)';`
	`$matches[1]`
	`True`
	`555-1212`

`(?! )`	Negatywna asercja z patrzeniem w przód. Gwarantuje, że dany wzorzec nie pasuje do dalszej części tekstu, ale nie przeprowadza rzeczywistego dopasowania.
	`PS >"przyjacielski" -match`
	`'(?!przyjacielski)przyjaciel'`
	`False`

`(?<= )`	Pozytywna asercja z patrzeniem w tył. Gwarantuje, że dany wzorzec pasuje do poprzedniej części tekstu, ale nie przeprowadza rzeczywistego dopasowania.
	`PS >"public int X" -match`
	`'^.*(?<=public )int .*$'`
	`True`

`(?<! )`	Negatywna asercja z patrzeniem w tył. Gwarantuje, że dany wzorzec nie pasuje do poprzedniej części tekstu, ale nie przeprowadza rzeczywistego dopasowania.
	`PS >"private int X" -match`
	`'^.*(?<!private )int .*$'`
	`False`

`(?> )`	Podwyrażenie bez cofania. Dopasowywane tylko wtedy, gdy może zostać dopasowane w całości.
	`PS >"Witaj, świecie" -match`
	`'(Witaj.*)wiecie'`
	`True`
	`PS >"Witaj, świecie" -match`
	`'(?>Witaj.*)wiecie'`
	`False`
	Wzorzec, który zawiera podwyrażenie bez cofania, nie zostaje dopasowany, ponieważ podwyrażenie dopasowuje cały łańcuch „Witaj, świecie".

Tabela 3.4. Niepodzielne asercje o zerowej szerokości: wzorce ograniczające miejsce, w którym tekst może zostać dopasowany

Asercja	Ograniczenie

^	Dopasowanie musi zachodzić na początku łańcucha (albo wiersza, jeśli włączona jest opcja `Multiline`).
	`PS >"Test" -match '^est'`
	`False`

$	Dopasowanie musi zachodzić na końcu łańcucha (albo wiersza, jeśli włączona jest opcja `Multiline`).
	`PS >"Test" -match 'Tes$'`
	`False`

\A	Dopasowanie musi zachodzić na początku łańcucha. `PS >"The`nTest" -match '(?m:^Test)'` `True` `PS >"The`nTest" -match '(?m:\ATest)'` `False`

\Z	Dopasowanie musi zachodzić na końcu łańcucha albo przed znakiem `\n` na końcu łańcucha.
	`PS >"Oto`nTest`n" -match '(?m:Oto$)'`
	`True`
	`PS >"Oto`nTest`n" -match '(?m:Oto\Z)'`
	`False`
	`PS >"Oto`nTest`n" -match 'Test\Z'`
	`True`

\z	Dopasowanie musi zachodzić na końcu łańcucha.
	`PS >"Oto`nTest`n" -match 'Test\z'`
	`False`

\G	Dopasowanie musi zachodzić tam, gdzie skończyło się poprzednie. Używana w połączeniu z metodą `System.Text.RegularExpressions.Match.NextMatch()`.

\b	Dopasowanie musi zachodzić na granicy słowa — pierwszych lub ostatnich znaków słowa rozdzielonych znakami niealfanumerycznymi.
	`PS >"Testowanie" -match 'nie\b'`
	`True`

\B	Dopasowanie nie może zachodzić na granicy słowa.
	`PS >"Testowanie" -match 'nie\B'`
	`False`

Tabela 3.5. Wzorce podstawienia: wzorce używane w operacjach zamiany tekstu

Wzorzec	Podstawienie

$*numer*	Tekst dopasowany przez grupę o numerze <*numer*>.
	`PS >"Test" -replace '(.*)st','$1ka'`
	`Teka`

${*nazwa*}	Tekst dopasowany przez grupę o nazwie <*nazwa*>.
	`PS >"Test" -replace`
	`'(?<pre>.*)st','${pre}ka'`
	`Teka`

$$	Znak $.
	`PS >"Test" -replace '.','$$'`
	`$$$$`

$&	Kopia całego dopasowania.
	`PS >"Test" -replace '^.*$','Znaleziono: $&'`
	`Znaleziono: Test`

$`	Tekst łańcucha wejściowego poprzedzający dopasowanie. `PS >"Test" -replace 'est$','Te$`'` `TteT`

$'	Tekst łańcucha wejściowego następujący po dopasowaniu.
	`PS >"Test" -replace '^Tes','Res$'''`
	`Restt`

$+	Ostatnia przechwycona grupa.
	`PS >"Testowanie" -replace '(.*)nie','$+ne'`
	`Testowane`

$_	Cały łańcuch wejściowy.
	`PS >"Testowanie" -replace`
	`'(.*)nie','Łańcuch: $_'`
	`Łańcuch: Testowanie`

Tabela 3.6. Konstrukcje alternacji: wyrażenia, które umożliwiają wykonywanie operacji albo-albo

Konstrukcja alternacji	Opis

`|`	Dopasowuje dowolny spośród elementów oddzielonych znakiem pionowej kreski.
	`PS >"Test" -match '(B|T)est'`
	`True`

`(?(`*wyrażenie*`)`
tak`|`*nie*`)`
	Dopasowuje element *tak*, jeśli *wyrażenie* pasuje w tym punkcie. W przeciwnym razie dopasowuje element *nie*. Element *nie* jest opcjonalny.
	`PS >"3.14" -match '(?(\d)3.14|Pi)'`
	`True`
	`PS >"Pi" -match '(?(\d)3.14|Pi)'`
	`True`
	`PS >"2.71" -match '(?(\d)3.14|Pi)'`
	`False`

`(?(`*nazwa*`)`
tak`|`*nie*`)`
	Dopasowuje element *tak*, jeśli grupa przechwytywania o nazwie *nazwa* pasuje w tym punkcie. W przeciwnym razie dopasowuje element *nie*. Element *nie* jest opcjonalny.
	`PS >"123" -match`
	`'(?<jeden>1)?(?(jeden)23|234)'`
	`True`
	`PS >"23" -match`
	`'(?<jeden>1)?(?(jeden)23|234)'`
	`False`
	`PS >"234" -match`
	`'(?<jeden>1)?(?(jeden)23|234)'`
	`True`

Tabela 3.7. Konstrukcje odwołania wstecznego: wyrażenia, które odwołują się do grupy przechwytywania w wyrażeniu

Konstrukcja odwołania wstecznego	Odwołuje się do

`\`*numer*	Grupy o numerze *numer*.
	`PS >"|Tekst|" -match '(.)Tekst\1'`
	`True`
	`PS >"|Tekst+" -match '(.)Tekst\1'`
	`False`

Tabela 3.7. Konstrukcje odwołania wstecznego: wyrażenia, które odwołują się do grupy przechwytywania w wyrażeniu (ciąg dalszy)

Konstrukcja odwołania wstecznego	Odwołuje się do

`\k<`*nazwa*`>`	Grupy o nazwie *nazwa*.
	`PS >"|Tekst|" -match '(?<Symbol>.)Tekst\k<Symbol>'`
	`True`
	`PS >"|Tekst+" -match '(?<Symbol>.)Tekst\k<Symbol>'`
	`False`

Tabela 3.8. Inne konstrukcje: pozostałe wyrażenia, które modyfikują wyrażenie regularne

Konstrukcja	Opis
`(?imnsx-imnsx: )`	Włącza lub wyłącza określoną opcję w pozostałej części wyrażenia. Oto obsługiwane opcje: `i` dopasowywanie bez uwzględniania wielkości liter `m` dopasowywanie wielowierszowe `n` jawne przechwytywanie `s` dopasowywanie jednowierszowe `x` ignorowanie odstępów `PS >"Te`nst" -match '(?sx)T e.st'` `True`

`(?# )`	Komentarz wpleciony w wyrażenie. Komentarze kończą się na pierwszym zamykającym nawiasie.
	`PS >"Test" -match '(?# Dopasowuje 'Test')Test'`
	`True`

`# [`*do końca wiersza*`]`	Postać komentarza dozwolona wtedy, gdy w wyrażeniu regularnym włączona jest opcja `IgnoreWhitespace`.
	`PS >"Test" -match '(?x)Test # Dopasowuje Test'`
	`True`

Tabela 3.9. Sekwencje unikowe: sekwencje znaków, które reprezentują inny znak

Sekwencja unikowa	Dopasowuje
`<zwykłe znaki>`	Znaki inne niż `. $ ^ { [ ( |) * + ? \` dopasowują same siebie.
`\a`	Dzwonek (alarm) `\u0007`.
`\b`	Znak cofania `\u0008`, jeśli znajduje się w klasie znakowej `[]`. W wyrażeniu regularnym `\b` oznacza granicę słowa (między znakami `\w` i `\W`) z wyjątkiem klasy znakowej `[]`, w której oznacza znak cofania. We wzorcu zastępowania `\b` zawsze oznacza znak cofania.
`\t`	Znak tabulacji `\u0009`.
`\r`	Znak powrotu karetki `\u000D`.
`\v`	Znak tabulacji pionowej `\u000B`.
`\f`	Znak wysuwu arkusza `\u000C`.
`\n`	Znak nowego wiersza `\u000A`.
`\e`	Znak ucieczki `\u001B`.
`\ddd`	Znak ASCII w notacji ósemkowej (do trzech cyfr). Liczby bez początkowego zera są traktowane jak odwołania wsteczne, jeśli mają tylko jedną cyfrę albo odpowiadają numerowi grupy przechwytywania.
`\xdd`	Znak ASCII w notacji szesnastkowej (dokładnie dwie cyfry).
`\cZ`	Znak kontrolny ASCII, na przykład `\cC` to *Control+C*.
`\udddd`	Znak Unicode w reprezentacji szesnastkowej (dokładnie cztery cyfry).
`\`	Jeśli poprzedza znak, który nie wchodzi w skład sekwencji unikowych, dopasowuje ten znak. Na przykład `\*` dopasowuje znak gwiazdki (`*`).