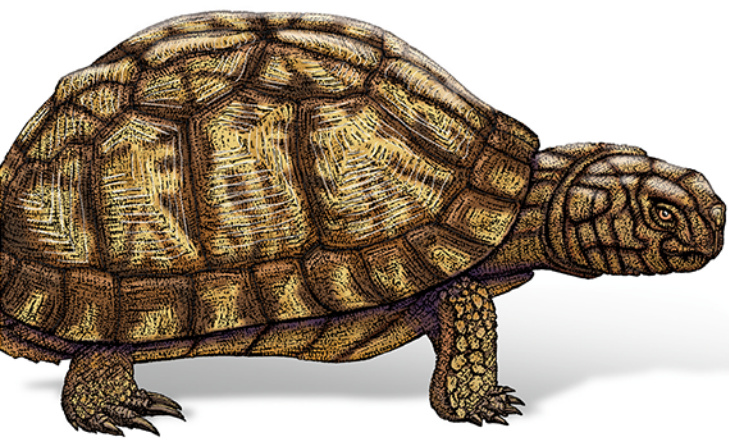
O'REILLY®

Wydanie III

PowerShell
Leksykon kieszonkowy

Przenośna pomoc
dla tworzących skrypty
w PowerShell



Helion

Lee Holmes

Tytuł oryginału: PowerShell Pocket Reference:
Portable Help for PowerShell Scripters, 3rd Edition

Tłumaczenie: Michał Sternik

ISBN: 978-83-283-8383-8

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/powlk3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

Rozdział 2.
Wyrażenia regularne

Wyrażenia regularne odgrywają ważną rolę w większości zadań analizowania i dopasowywania tekstu. Stanowią one ważną podstawę dla operatorów `-split` i `-match`, instrukcji `switch`, polecenia cmdlet `Select-String` i innych. W tabelach od 2.1 do 2.10 umieszczono listy często używanych wyrażeń regularnych.

Tabela 2.1. Klasy znaków — wzorce reprezentujące zestawy znaków

Klasa znaków	Dopasowanie
.	Dowolny znak z wyjątkiem nowego wiersza. Jeśli wyrażenie regularne używa atrybutu `SingleLine`, pasuje do dowolnego znaku. PS > "T" -match '.' True
[*znaki*]	Dowolny znak z tych, które zawarto w nawiasach. Na przykład: `[aeiou]`. PS > "Test" -match '[Tes]' True
[^*znaki*]	Dowolny znak oprócz tych, które zawarto w nawiasach. Na przykład: `[^aeiou]`. PS > "Test" -match '[^Tes]' False
[*początek-koniec*]	Dowolny znak między pierwszym a drugim podanym znakiem włącznie. W nawiasie może się znajdować wiele zakresów znaków. Na przykład `[a-eh-j]`. PS > "Test" -match '[e-t]' True

Klasa znaków	Dopasowanie
[^*początek-koniec*]	Dowolny znak oprócz tych, które znajdują się w podanych zakresach znaków włącznie. Między nawiasami może się znajdować wiele zakresów znaków. Na przykład [^a-eh-j].
	PS > "Test" -match '[^e-t]'
	False
\p{*klasa znaków*}	Dowolny znak w grupie Unicode lub zakresie blokowym określonym przez {*klasa znaków*}.
	PS > "+" -match '\p{Sm}'
	True
\P{*klasa znaków*}	Dowolny znak poza grupą znaków Unicode lub poza zakresem blokowym określonym przez {*klasa znaków*}.
	PS > "+" -match '\P{Sm}'
	False
\w	Dowolny znak słowa. Należy zauważyć, że w definicji Unicode znak słowa zawiera cyfry, a także wiele symboli matematycznych i różnych innych symboli.
	PS > "a" -match '\w'
	True
\W	Każdy znak niebędący znakiem słowa.
	PS > "!" -match '\W'
	True
\s	Dowolny znak odstępu.
	PS > "`t" -match '\s'
	True
\S	Dowolny znak poza znakiem odstępu.
	PS > " `t" -match '\S'
	False
\d	Dowolna cyfra dziesiętna.
	PS > "5" -match '\d'
	True
\D	Dowolny znak, który nie jest cyfrą dziesiętną.
	PS > "!" -match '\D'
	True

Tabela 2.2. Kwantyfikatory — wyrażenia wymuszające liczbę znaków w poprzednim wyrażeniu

Kwantyfikator	Znaczenie
<none>	Jedno dopasowanie. PS > "T" -match 'T' True
*	Zero lub więcej dopasowań, dopasowuje jak najwięcej. PS > "A" -match 'T*' True PS > "TTTTT" -match '^T*$' True PS > 'ATTT' -match 'AT*'; $Matches[0] True ATTT
+	Jedno lub więcej dopasowań, dopasowuje jak najwięcej. PS > "A" -match 'T+' False PS > "TTTTT" -match '^T+$' True PS > 'ATTT' -match 'AT+'; $Matches[0] True ATTT
?	Zero lub jedno dopasowanie, dopasowuje jak najwięcej. PS > "TTTTT" -match '^T?$' False PS > 'ATTT' -match 'AT?'; $Matches[0] True AT
{*n*}	Dokładnie *n* dopasowań. PS > "TTTTT" -match '^T{5}$' True

Kwantyfikator	Znaczenie
{*n*,}	*n* lub więcej dopasowań, dopasowuje jak najwięcej. PS > "TTTTT" -match '^T{4,}$' True
{*n*,*m*}	Pomiędzy *n* a *m* dopasowań (włącznie), dopasowuje jak najwięcej. PS > "TTTTT" -match '^T{4,6}$' True
*?	Zero lub więcej dopasowań, dopasowuje jak najmniej. PS > "A" -match '^AT*?$' True PS > 'ATTT' -match 'AT*?'; $Matches[0] True A
+?	Jedno lub więcej dopasowań, dopasowuje jak najmniej. PS > "A" -match '^AT+?$' False PS > 'ATTT' -match 'AT+?'; $Matches[0] True AT
??	Zero dopasowań lub jedno dopasowanie, dopasowuje jak najmniej. PS > "A" -match '^AT??$' True PS > 'ATTT' -match 'AT??'; $Matches[0] True A
{*n*}?	Dokładnie *n* dopasowań. PS > "TTTTT" -match '^T{5}?$' True

Kwantyfikator	Znaczenie
{*n*,}?	*n* lub więcej dopasowań, dopasowuje jak najmniej. PS > "TTTTT" -match '^T{4,}?$' True
{*n*,*m*}?	Między *n* a *m* dopasowań (włącznie), dopasowuje jak najmniej. PS > "TTTTT" -match '^T{4,6}?$' True

Tabela 2.3. Grupowanie konstruktów — wyrażenia, które umożliwiają grupowanie znaków, wzorców i innych wyrażeń

Konstrukt grupowania	Opis
(*tekst*)	Przechwytuje tekst dopasowany wewnątrz nawiasów. Przechwytywania są przypisane do liczby (począwszy od jedynki) na podstawie kolejności w nawiasach. PS > "Hello" -match '^(.*)llo$'; $matches[1] True He
(?<*nazwa*>)	Przechwytuje tekst dopasowany wewnątrz nawiasów. Te przechwytywania otrzymują nazwę podaną w nawiasie. PS > "Hello" -match '^(?<One>.*)llo$'; $matches.One True He
(?<*nazwa1*-*nazwa2*>)	Definicja grupy równoważącej. Jest to zaawansowana konstrukcja wyrażenia regularnego, która pozwala dopasować równomiernie zrównoważone pary terminów.
(?:)	Grupa nieprzechwytująca. PS > "A1" -match '((A|B)\d)'; $matches True Name Value ---- -----

Konstrukt grupowania	Opis
	2 A 1 A1 0 A1 PS > "A1" -match '((A|B)\d)'; $matches True Name Value ---- ----- 2 A 1 A1 0 A1
(?*imnsx*-*imnsx*:)	Stosuje lub wyłącza daną opcję dla tej grupy. Obsługiwane opcje to: *i — nieuwzględnianie wielkości liter* *m — wielowierszowe* *n — dokładne przechwytywanie* *s — jednowierszowe* *x — ignorowanie odstępów* PS > "Te`nst" -match '(T e.st)' False PS > "Te`nst" -match '(?sx:T e.st)' True
(?=)	Pozytywna asercja wyprzedzająca o zerowej szerokości. Zapewnia, że dany wzorzec pasuje do wyrażenia z prawej strony, bez faktycznego wykonywania dopasowania. PS > "555-1212" -match '(?=...-)(.*)'; $matches[1] True 555-1212
(?!)	Negatywna asercja wsteczna o zerowej szerokości. Zapewnia, że dany wzorzec nie pasuje do wyrażenia z prawej strony, bez faktycznego wykonywania dopasowania. PS > "przyjacielski" -match '(?!przyjacielski)przyjaciel' False

Tabela 2.4. Więcej konstrukcji grupowania

Konstrukt grupowania	Opis
(?<=)	Pozytywna asercja wsteczna o zerowej szerokości. Zapewnia, że dany wzorzec pasuje do wyrażenia z lewej strony, bez faktycznego wykonywania dopasowania. PS > "public int X" -match '^.*(?<=public)int .*$' True
(?<!)	Negatywna asercja wsteczna o zerowej szerokości. Zapewnia, że dany wzorzec nie pasuje do wyrażenia z lewej strony, bez faktycznego wykonywania dopasowania. PS > "private int X" -match '^.*(?<!private)int .*$' False
(?>)	Podwyrażenie bez cofania. Dopasowuje tylko wtedy, gdy to podwyrażenie można dopasować całkowicie. PS > "Hello World" -match '(Hello.*)orld' True PS > "Hello World" -match '(?>Hello.*)orld' False Wersja podwyrażenia bez cofania nie powoduje dopasowania, ponieważ jego pełne dopasowanie to `"Hello World"`.

Tabela 2.5. Niepodzielne asercje o zerowej szerokości — wzorce ograniczające miejsce, w którym może wystąpić dopasowanie

Asercja	Ograniczenie
^	Dopasowanie musi wystąpić na początku ciągu znaków (lub wiersza, jeśli użyto opcji `Multiline`). PS > "Test" -match '^est' False
$	Dopasowanie musi wystąpić na końcu ciągu znaków (lub wiersza, jeśli użyto opcji `Multiline`). PS > "Test" -match 'Tes$' False

Asercja	Ograniczenie
`\A`	Dopasowanie musi wystąpić na początku ciągu znaków.
	PS > "The`nTest" -match '(?m:^Test)'
	True
	PS > "The`nTest" -match '(?m:\ATest)'
	False
`\Z`	Dopasowanie musi wystąpić na końcu ciągu znaków lub przed znakiem nowego wiersza `\n`.
	PS > "The`nTest`n" -match '(?m:The$)'
	True
	PS > "The`nTest`n" -match '(?m:The\Z)'
	False
	PS > "The`nTest`n" -match 'Test\Z'
	True
`\z`	Dopasowanie musi wystąpić na końcu ciągu znaków.
	PS > "The`nTest`n" -match 'Test\z'
	False
`\G`	Dopasowanie musi nastąpić w miejscu zakończenia poprzedniego dopasowania. Używany z `System.Text.RegularExpressions.Match.NextMatch()`.
`\b`	Dopasowanie musi mieć miejsce na granicy słowa: pierwszy lub ostatni znak w słowach oddzielonych znakami niealfanumerycznymi.
	PS > "Testing" -match 'ing\b'
	True
`\B`	Dopasowanie nie może wystąpić na granicy słowa.
	PS > "Testing" -match 'ing\B'
	False

Tabela 2.6. Wzorce podstawiania — wzorce używane w operacji zastępowania wyrażenia regularnego

Wzorzec	Cel zastępowania
`$number`	Tekst dopasowany do numeru grupy.
	PS > "Test" -replace "(.*)st",'$1ar'
	Tear
`${name}`	Tekst dopasowany według nazwy grupy.
	PS > "Test" -replace "(?<pre>.*)st",'${pre}ar'
	Tea
$$	Dosłownie $.
	PS > "Test" -replace ".",'$$'
	$$$$
$&	Kopia całego dopasowania.
	PS > "Test" -replace "^.*$",'Found: $&'
	Found: Test
$`	Tekst wejściowego ciągu znaków poprzedzającego dopasowanie.
	PS > "Test" -replace "est$",'Te$`'
	TTeT
$'	Tekst wejściowego ciągu znaków, który następuje po dopasowaniu.
	PS > "Test" -replace "^Tes",'Res$'"
	Restt
$+	Ostatnia przechwycona grupa.
	PS > "Testing" -replace "(.*)ing",'$+ed'
	Tested
$_	Cały wejściowy ciąg znaków.
	PS > "Testing" -replace "(.*)ing",'String: $_'
	String: Testing

Tabela 2.7. Konstrukcje alternacji — wyrażenia, które umożliwiają wykonywanie logiki i/lub

Konstrukt modyfikacji	Opis
|	Dopasowuje dowolny z terminów oddzielonych znakiem pionowej kreski. PS > "Test" -match '(B|T)est' True
(?(*wyrażenie*) *tak*|*nie*)	Dopasowuje termin `tak`, jeśli wyrażenie pasuje w tym miejscu. W przeciwnym razie dopasowuje do terminu `nie`. Termin `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 termin `tak`, jeśli grupa przechwytywania o nazwie `nazwa` pasuje w tym miejscu. W przeciwnym razie dopasowuje do terminu `nie`. Termin `nie` jest opcjonalny. PS > "123" -match '(?<one>1)?(?(one)23|234)' True PS > "23" -match '(?<one>1)?(?(one)23|234)' False PS > "234" -match '(?<one>1)?(?(one)23|234)' True

Tabela 2.8. Konstrukty wnioskowania wstecznego — wyrażenia odwołują się do grupy przechwytywania w wyrażeniu

Konstrukt wnioskowania wstecznego	Odnosi się do
numer	Numer grupy *numer* w wyrażeniu. PS > "|Text|" -match '(.)Text\1' True PS > "|Text+" -match '(.)Text\1' False
\k<*nazwa*>	Grupa o nazwie *nazwa* w wyrażeniu. PS > "|Text|" -match '(?<Symbol>.)Text\k<Symbol>' True PS > "|Text+" -match '(?<Symbol>.)Text\k<Symbol>' False

Tabela 2.9. Inne konstrukty — inne wyrażenia modyfikujące wyrażenie regularne

Konstrukt	Opis
(?*imnsx*-*imnsx*)	Stosuje lub wyłącza daną opcję dla pozostałej części tego wyrażenia. Obsługiwane opcje to: *i — nieuwzględnianie wielkości liter* *m — wielowierszowe* *n — dokładne przechwytywanie* *s — jednowierszowe* *x — ignorowanie odstępów* PS > "Te`nst" -match '(?sx)T e.st' True
(?#)	Komentarz w wierszu. Kończy się to przy pierwszym zamykającym nawiasie. PS > "Test" -match '(?# Match "Test")Test' True

Konstrukt	Opis
`# [do końca wiersza]`	Forma komentarza jest dozwolona, gdy wyrażenie regularne ma włączoną opcję `IgnoreWhitespace`. PS > "Test" -match '(?x)Test # Matches Test' True

Tabela 2.10. Znaki ucieczki — sekwencje znaków, które reprezentują inny znak

Znak ucieczki	Dopasowanie
<zwykłe znaki>	Dopasowanie znaków innych niż `. $ ^ { [ ( | ) * + ? \`.
`\a`	Dzwonek (alarm) `\u0007`.
`\b`	Znak backspace `\u0008`, jeśli występuje w nawiasie `[]` oznaczającym klasę znaków. W wyrażeniu regularnym `\b` oznacza granicę wyrazu (między znakami `\w` i `\W`) z wyjątkiem nawiasu `[]`, który oznacza klasy znaków, gdzie `\b` odnosi się do znaku backspace. We wzorcu zastępczym `\b` zawsze oznacza backspace.
`\t`	Znak tabulacji `\u0009`.
`\r`	Znak powrotu karetki `\u000D`.
`\v`	Tabulator pionowy `\u000B`.
`\f`	Kanał informacyjny formularza `\u000C`.
`\n`	Znak nowego wiersza `\u000A`.
`\e`	Znak ucieczki `\u001B`.
`\`*ddd*	Znak ASCII jako ósemkowy (do trzech cyfr). Liczby bez zera wiodącego są traktowane jako referencje wsteczne, jeśli mają tylko jedną cyfrę lub jeśli odpowiadają numerowi grupy przechwytywania.
`\x`*dd*	Znak ASCII używający reprezentacji szesnastkowej (dokładnie dwie cyfry).
`\c`*C*	Znak sterujący ASCII; na przykład `\cC` oznacza `Control-C`.
`\u`*dddd*	Znak Unicode przy użyciu reprezentacji szesnastkowej (dokładnie cztery cyfry).
`\`	Gdy poprzedza znak, który nie jest rozpoznawany jako znak ucieczki, powoduje dopasowanie do tego znaku. Na przykład `\*` oznacza dosłownie znak *.


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

PowerShell: to, co adminowi pomoże najlepiej!

PowerShell powstał w 2006 roku. Jest to potężne i uniwersalne narzędzie, pomocne w automatyzacji niemal wszystkich żmudnych czynności, które administrator musi często wykonywać. Aby nie tracić czasu na powtarzalne zadania, trzeba tylko umieć biegle pisać skrypty PowerShell. Przyswojenie tej umiejętności jest łatwiejsze, jeśli wykorzystuje się logikę obiektową PowerShell, a także to, że obiekt powstający po wykonaniu polecenia ma swoje metody i właściwości.

Ta książka jest zwięzłym przewodnikiem po programie PowerShell. Zaprezentowano tu praktyczne podstawy języka PowerShell: operatory, instrukcje warunkowe, pętle, zasady pracy na ciągach znaków. Opisano procedurę tworzenia i uruchamiania skryptów. Pokazano, jak wprowadzać dane wejściowe do poleceń, dostosowywać zachowanie poleceń do konkretnych sytuacji, a także zarządzać błędami. Przydatną częścią publikacji są wskazówki, jak tworzyć polecenia, funkcje i bloki skryptów.

W książce między innymi:

- wyrażenia regularne
- formatowanie ciągów znaków i obiektu DateTime przy użyciu .NET
- wybrane klasy .NET i ich zastosowania
- podręczna dokumentacja WMI
- wybrane obiekty COM i ich zastosowania

Lee Holmes jest głównym architektem zabezpieczeń w Azure Security. Brał udział w tworzeniu programu PowerShell od jego wersji beta. Ma ogromne doświadczenie w pracy z Windows PowerShell, jest też niezastąpiony w rozwiązywaniu problemów administratorów systemów i użytkowników tego programu.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-8383-8

INFORMATYKA W NAJLEPSZYM WYDANIU
Cena: 39,90 zł