.NET FRAMEWORK REGULAR EXPRESSIONS

SINGLE CHARACTERS

Use	To match any character
[*set***]**	In that set
[^*set***]**	Not in that set
[*a–z***]**	In the *a-z* range
[^*a–z***]**	Not in the a-z range
.	Any except \n (new line)
*****char*	Escaped special character

CONTROL CHARACTERS

Use	To match	Unicode
\t	Horizontal tab	\u0009
\v	Vertical tab	\u000B
\b	Backspace	\u0008
\e	Escape	\u001B
\r	Carriage return	\u000D
\f	Form feed	\u000C
\n	New line	\u000A
\a	Bell (alarm)	\u0007
\c *char*	ASCII control character	–

NON-ASCII CODES

Use	To match character with
*****octal*	2-3 digit octal character code
\x *hex*	2-digit hex character code
\u *hex*	4-digit hex character code

CHARACTER CLASSES

Use	To match character
\p{*ctgry***}**	In that Unicode category or block
\P{*ctgry***}**	Not in that Unicode category or block
\w	Word character
\W	Non-word character
\d	Decimal digit
\D	Not a decimal digit
\s	White-space character
\S	Non-white-space char

QUANTIFIERS

Greedy	Lazy	Matches
*	*?	0 or more times
+	+?	1 or more times
?	??	0 or 1 time
{*n*}	{***n***}?	Exactly *n* times
{*n*,}	{***n***,}?	At least *n* times
{*n*,*m*}	{***n***,***m***}?	From *n* to *m* times

ANCHORS

Use	To specify position
^	At start of string or line
\A	At start of string
\z	At end of string
\Z	At end (or before \n at end) of string
$	At end (or before \n at end) of string or line
\G	Where previous match ended
\b	On word boundary
\B	Not on word boundary

GROUPS

Use	To define
(*exp***)**	Indexed group
(?<*name***>***exp***)**	Named group
(?<*name1***-***name2***>***exp***)**	Balancing group
(?:*exp***)**	Noncapturing group
(?=*exp***)**	Zero-width positive lookahead
(?!*exp***)**	Zero-width negative lookahead
(?<=*exp***)**	Zero-width positive lookbehind
(?<!*exp***)**	Zero-width negative lookbehind
(?>*exp***)**	Non-backtracking (greedy)

INLINE OPTIONS

Option	Effect on match
i	Case-insensitive
m	Multiline mode
n	Explicit (named)
s	Single-line mode
x	Ignore white space

Use	To
(?imnsx-imnsx)	Set or disable the specified options
(?imnsx-imnsx:*exp***)**	Set or disable the specified options within the expression

June 2014

BACKREFERENCES

Use	To match
*****n*	Indexed group
\k<*name***>**	Named group

ALTERNATION

Use	To match
a **|** *b*	Either *a* or *b*
(?(*exp***)** *yes* **|** *no***)**	*yes* if *exp* is matched *no* if *exp* isn't matched
(?(*name***)** *yes* **|** *no***)**	*yes* if *name* is matched *no* if *name* isn't matched

SUBSTITUTION

Use	To substitute
$*n*	Substring matched by group number *n*
${*name***}**	Substring matched by group *name*
$$	Literal $ character
$&	Copy of whole match
$`	Text before the match
$'	Text after the match
$+	Last captured group
$_	Entire input string

COMMENTS

Use	To
(?# *comment***)**	Add inline comment
#	Add x-mode comment

For detailed information and examples, see **http://aka.ms/regex**

To test your regular expressions, see **http://regexlib.com/RETester.aspx**

SUPPORTED UNICODE CATEGORIES

Category	Description
Lu	Letter, uppercase
Ll	Letter, lowercase
Lt	Letter, title case
Lm	Letter, modifier
Lo	Letter, other
L	Letter, all
Mn	Mark, nonspacing combining
Mc	Mark, spacing combining
Me	Mark, enclosing combining
M	Mark, all diacritic
Nd	Number, decimal digit
Nl	Number, letterlike
No	Number, other
N	Number, all
Pc	Punctuation, connector
Pd	Punctuation, dash
Ps	Punctuation, opening mark
Pe	Punctuation, closing mark
Pi	Punctuation, initial quote mark
Pf	Puntuation, final quote mark
Po	Punctuation, other
P	Punctuation, all
Sm	Symbol, math
Sc	Symbol, currency
Sk	Symbol, modifier
So	Symbol, other
S	Symbol, all
Zs	Separator, space
Zl	Separator, line
Zp	Separator, paragraph
Z	Separator, all
Cc	Control code
Cf	Format control character
Cs	Surrogate code point
Co	Private-use character
Cn	Unassigned
C	Control characters, all

For named character set blocks (e.g., Cyrillic), search for "supported named blocks" in the MSDN Library.

REGULAR EXPRESSION OPERATIONS

Class: System.Text.RegularExpressions.Regex

Pattern matching with Regex objects

To initialize with	Use constructor
Regular exp	Regex(String)
+ options	Regex(String, RegexOptions)
+ time-out	Regex(String, RegexOptions, TimeSpan)

Pattern matching with static methods

Use an overload of a method below to supply the regular expression and the text you want to search.

Finding and replacing matched patterns

To	Use method
Validate match	Regex.IsMatch
Retrieve single match	Regex.Match (first) Match.NextMatch (next)
Retrieve all matches	Regex.Matches
Replace match	Regex.Replace
Divide text	Regex.Split
Handle char escapes	Regex.Escape Regex.Unescape

Getting info about regular expression patterns

To get	Use Regex API
Group names	GetGroupNames GetGroupNameFromNumber
Group numbers	GetGroupNumbers GetGroupNumberFromName
Expression	ToString
Options	Options
Time-out	MatchTimeOut
Cache size	CacheSize
Direction	RightToLeft

Para ver el artículo en inglés, active la casilla Inglés. También puede ver el texto en inglés en una ventana emergente si pasa el puntero del mouse por el texto.

Lenguaje de expresiones regulares - Referencia rápida

.NET Framework (current version)

Una expresión regular es un modelo con el que el motor de expresiones regulares intenta buscar una coincidencia en el texto de entrada. Un modelo consta de uno o más literales de carácter, operadores o estructuras. Para obtener una breve introducción, consulte Expresiones regulares de .NET Framework.

Cada sección de esta referencia rápida enumera una categoría determinada de caracteres, operadores y construcciones que puede usar para definir expresiones regulares:

Esta información también se proporciona en dos formatos que se puede descargar e imprimir para facilitar su consulta:

Descargar en formato Word (.docx)
Descargar en formato PDF (.pdf)

Escapes de carácter

El carácter de barra diagonal inversa (\) en una expresión regular indica que el carácter que lo sigue es un carácter especial (como se muestra en la tabla siguiente) o que se debe interpretar literalmente. Para obtener más información, véase Novedades en el SDK Visual Studio 2015.

Carácter de escape	Descripción	Modelo	Coincidencias
\a	Coincide con un carácter de campana, \u0007.	`\a`	"\u0007" en "¡Error!" + '\u0007'
\b	En una clase de caracteres, coincide con un retroceso, \u0008.	`[\b]{3,}`	"\b\b\b\b" en "\b\b\b\b"
\t	Coincide con una tabulación, \u0009.	`(\w+)\t`	"artículo1\t", "artículo2\t" en "artículo1\tartículo2\t"
\r	Coincide con un retorno de carro, \u000D. (**Archivo**, **nuevo**, **proyecto**, o tipo **proyecto** en la ventana de **Inicio rápido**).	`\r\n(\w+)`	"\r\nEstas" en "\r\nEstas son\ndos líneas."
\v	Coincide con una tabulación vertical, \u000B.	`[\v]{2,}`	"\v\v\v" en "\v\v\v"
\f	Coincide con un avance de página, \u000C.	`[\f]{2,}`	"\f\f\f" en "\f\f\f"
\n	Coincide con una nueva línea, \u000A.	`\r\n(\w+)`	"\r\nEstas" en "\r\nEstas son\ndos líneas."
\e	Coincide con un escape, \u001B.	`\e`	"\x001B" en "\x001B"
*****nnn*	Usa la representación octal para especificar un carácter (*nnn* consta de dos o tres dígitos).	`\w\040\w`	"a b", "c d" en "a bc d"
\x*nn*	Usa la representación hexadecimal para especificar un carácter (*nn* consta de exactamente dos dígitos).	`\w\x20\w`	"a b", "c d" en "a bc d"
\c*X* **\c***x*	Coincide con el carácter de control ASCII especificado por *X* o *x*, donde *X* o *x* es la letra del carácter de control.	`\cC`	"\x0003" en "\x0003" (Ctrl-C)
\u*nnnn*	Coincide con un carácter Unicode usando la representación hexadecimal (exactamente cuatro dígitos, según representa *nnnn*).	`\w\u0020\w`	"a b", "c d" en "a bc d"
****	Cuando va seguido de un carácter que no se reconoce como un carácter de escape en esta y otras tablas de este tema, coincide con ese carácter. Por ejemplo, ***** es igual que **\x2A** y **\.** es igual que **\x2E**. Esto permite que el motor de expresiones regulares elimine la ambigüedad de los elementos del lenguaje (como * o ?) y los literales de carácter (representados por * o \?).	`\d+[\+-x\*]\d+`	"2+2" y "3*9" en "(2+2) * 3*9"

Volver al principio

Clases de carácter

Una clase de caracteres coincide con cualquiera de un juego de caracteres. Las clases de caracteres incluyen los elementos del lenguaje enumerados en la tabla siguiente. Para obtener más información, véase Novedades en el SDK Visual Studio

Clase de carácter	Descripción	Modelo	Coincidencias
[*grupo_caracteres***]**	Coincide con cualquier carácter individual de *grupo_caracteres*. De forma predeterminada, la coincidencia distingue entre mayúsculas y minúsculas.	`[ae]`	"a" en "casa" "a", "e" en "ave"
[^*grupo_caracteres***]**	Negación: coincide con cualquier carácter individual que no esté en *grupo_caracteres*. De forma predeterminada, los caracteres de *grupo_caracteres* distinguen entre mayúsculas y minúsculas.	`[^aei]`	"r", "n", "o" en "reino"
[*first***-***last***]**	Intervalo de caracteres: coincide con cualquier carácter individual en el intervalo de *primero* a *último*.	`[A-Z]`	"A", "B" en "AB123"
.	Carácter comodín: coincide con cualquier carácter excepto con \n. Para coincidir con un carácter de punto literal (. o **\u002E**), debe anteponerle el carácter de escape (**\.**).	`a.e`	"ave" en "llave" "ate" en "yate"

\p{ *name* **}**	Coincide con cualquier carácter individual que pertenezca a la categoría general Unicode o al bloque con nombre especificado por *name*.	`\p{Lu}`
		
		`\p{IsCyrillic}`
			"C", "L" en "City Lights"
			
			"Д", "Ж" in "ДЖem"

\P{ *name* **}**	Coincide con cualquier carácter individual que no pertenezca a la categoría general Unicode o al bloque con nombre especificado por *name*.	`\P{Lu}`
		
		`\P{IsCyrillic}`
			"i", "t", "y" en "City"
			
			"e", "m" in "ДЖem"

\w	Coincide con cualquier carácter de una palabra.	`\w`	"I", "D", "A", "1", "3" en "ID A1.3"
\W	Coincide con cualquier carácter que no pertenezca a una palabra.	`\W`	" ", "." en "ID A1.3"
\s	Coincide con cualquier carácter que sea un espacio en blanco.	`\w\s`	"D " en "ID A1.3"
\S	Coincide con cualquier carácter que no sea un espacio en blanco.	`\s\S`	" _" en "int __ctr"
\d	Coincide con cualquier dígito decimal.	`\d`	"4" en "4 = IV"
\D	Coincide con cualquier carácter que no sea un dígito decimal.	`\D`	" ", "=", " ", "I", "V" en "4 = IV"

Volver al principio

Delimitadores

Los delimitadores, o aserciones atómicas de ancho cero, hacen que una coincidencia tenga éxito o no dependiendo de la posición actual en la cadena, pero no hacen que el motor avance por la cadena ni consuma caracteres. Los metacaracteres enumerados en la tabla siguiente son delimitadores. Para obtener más información, véase Novedades en el SDK Visual Studio 2015.

Aserción	Descripción	Modelo	Coincidencias
^	La coincidencia debe comenzar al principio de la cadena o de la línea.	`^\d{3}`	"901" en "901-333-" "901-333-"
$	La coincidencia se debe producir al final de la cadena o antes de **\n** al final de la línea o de la cadena.	`-\d{3}$`	"-333" en "-901-333" "-901-333"
\A	La coincidencia se debe producir al principio de la cadena.	`\A\d{3}`	"901" en "901-333-" "901-333-"
\Z	La coincidencia se debe producir al final de la cadena o antes de **\n** al final de la cadena.	`-\d{3}\Z`	"-333" en "-901-333" "-901-333"
\z	La coincidencia se debe producir al final de la cadena.	`-\d{3}\z`	"-333" en "-901-333" "-901-333"
\G	La coincidencia se debe producir en el punto en el que finalizó la coincidencia anterior.	`\G\(\d\)`	"(1)", "(3)", "(5)" en "(1)(3)(5)[7](9)"
\b	La coincidencia se debe producir en un límite entre un carácter **\w** (alfanumérico) y un carácter **\W** (no alfanumérico).	`\b\w+\s\w+\b`	"ellos ello", "ellos ellos" en "ellos ello ellos ellos"
\B	La coincidencia no se debe producir en un límite **\b**.	`\Bend\w*\b`	"fin", "final" en "finalizar finalista finalizador finalizó"

Volver al principio

Construcciones de agrupamiento

Las construcciones de agrupamiento definen subexpresiones de una expresión regular y, normalmente, capturan subcadenas de una cadena de entrada. Las construcciones de agrupamiento incluyen los elementos del lenguaje enumerados en la tabla siguiente. Para obtener más información, véase Novedades en el SDK Visual Studio 2015.

Construcción de agrupamiento	Descripción	Modelo	Coincidencias
(*subexpresión***)**	Captura la subexpresión coincidente y le asigna un número ordinal basado en uno.	`(\w)\1`	"aa" en "aarón"
(?< *nombre***>** *subexpresión***)**	Captura la subexpresión coincidente en un grupo con nombre.	`(?<double>\w)\k<double>`	"aa" en "aarón"
(?< *nombre1***-** *nombre2***>** *subexpresión***)**	Define una definición de grupo de equilibrio. Para obtener más información, consulte la sección "Definiciones de grupos de equilibrio" en Construcciones de agrupamiento en expresiones regulares.	`(((?'Open'\()[^\(\)]*)+((?'Close-Open'\))[^\(\)]*)+)*(?(Open)(?!))$`	"((1-3)*(3-1))" en "3+2^((1-3)*(3-1))"
(?: *subexpresión***)**	Define un grupo sin captura.	`Write(?:Line)?`	"WriteLine" en "Console.WriteLine()" "Write" en "Console.Write(valor)"
(?imnsx-imnsx: *subexpresión***)**	Aplica o deshabilita las opciones especificadas dentro de *subexpresión*. Para obtener más información, véase Novedades en el SDK Visual Studio 2015.	`A\d{2}(?i:\w+)\b`	"A12xI", "A12XL" en "A12xI A12XL a12xI"
(?= *subexpresión***)**	Aserción de búsqueda anticipada positiva de ancho cero.	`\w+(?=\.)`	"es", "corría" y "hermoso" en "Él es. El perro corría. El sol está hermoso."
(?! *subexpresión***)**	Aserción de búsqueda anticipada negativa de ancho cero.	`\b(?!un)\w+\b`	"seguro", "usado" en "aseguro seguro unidad usado"
(?<= *subexpresión***)**	Aserción de búsqueda tardía positiva de ancho cero.	`(?<=19)\d{2}\b`	"99", "50", "05" en "1851 1999 1950 1905 2003"
(?<! *subexpresión***)**	Aserción de búsqueda tardía negativa de ancho cero.	`(?<!19)\d{2}\b`	"51", "03" en "1851 1999 1950 1905 2003"
(?> *subexpresión***)**	Subexpresión sin retroceso (o "expansiva").	`[13579](?>A+B+)`	"1ABB", "3ABB" y "5AB" en "1ABB 3ABBC 5AB 5AC"

Volver al principio

Cuantificadores

Un cuantificador especifica cuántas instancias del elemento anterior (que puede ser un carácter, un grupo o una clase de caracteres) debe haber en la cadena de entrada para que se encuentre una coincidencia. Los cuantificadores incluyen los elementos del lenguaje enumerados en la tabla siguiente. Para obtener más información, véase Novedades en el SDK Visual Studio 2015.

Cuantificador	Descripción	Modelo	Coincidencias
*****	Coincide con el elemento anterior cero o más veces.	`\d*\.\d`	".0", "19.9", "219.9"
+	Coincide con el elemento anterior una o más veces.	`"be+"`	"caí" en "caída", "be" en "bebé"
?	Coincide con el elemento anterior cero veces o una vez.	`"rai?n"`	"rata", "raicilla"
{*n***}**	Coincide con el elemento anterior exactamente *n* veces.	`",\d{3}"`	",043" en "1,043.6", ",876", ",543", y ",210" en "9,876,543,210"
{*n***,}**	Coincide con el elemento anterior al menos *n* veces.	`"\d{2,}"`	"166", "29", "1930"
{*n***,***m***}**	Coincide con el elemento anterior al menos *n* veces, pero no más de *m* veces.	`"\d{3,5}"`	"166", "17668" "19302" en "193024"
***?**	Coincide con el elemento anterior cero o más veces, pero el menor número de veces que sea posible.	`\d*?\.\d`	".0", "19.9", "219.9"
+?	Coincide con el elemento anterior una o más veces, pero el menor número de veces que sea posible.	`"be+?"`	"be" en "bebida", "be" en "bebé"
??	Coincide con el elemento anterior cero o una vez, pero el menor número de veces que sea posible.	`"rai??n"`	"rata", "raicilla"
{*n***}?**	Coincide con el elemento precedente exactamente *n* veces.	`",\d{3}?"`	",043" en "1,043.6", ",876", ",543", y ",210" en "9,876,543,210"
{*n***,}?**	Coincide con el elemento anterior al menos *n* veces, pero el menor número de veces posible.	`"\d{2,}?"`	"166", "29", "1930"
{*n***,***m***}?**	Coincide con el elemento anterior entre *n* y *m* veces, pero el menor número de veces posible.	`"\d{3,5}?"`	"166", "17668" "193", "024" en "193024"

Volver al principio

Construcciones de referencia inversa

Una referencia inversa permite identificar una subexpresión coincidente previamente más adelante en la misma expresión regular. En la tabla siguiente se enumeran las construcciones de referencia inversa admitidas en las expresiones regulares de .NET Framework. Para obtener más información, véase Novedades en el SDK Visual Studio 2015.

Construcción de referencias inversas	Descripción	Modelo	Coincidencias
****_número_	Referencia inversa. Coincide con el valor de una subexpresión numerada.	`(\w)\1`	"aa" en "aarón"
\k< _name_ **>**	Referencia inversa con nombre Coincide con el valor de una expresión con nombre.	`(?<char>\w)\k<char>`	"aa" en "aarón"

Volver al principio

Construcciones de alternancia

Las estructuras de alternancia modifican una expresión regular para habilitar o no la coincidencia. Estas construcciones incluyen los elementos del lenguaje enumerados en la tabla siguiente. Para obtener más información, véase Novedades en el SDK Visual Studio 2015.

Construcciones de alternancia	Descripción	Modelo	Coincidencias
|	Coincide con cualquier elemento separado por el carácter de barra vertical (|).	`th(e|is|at)`	"el", "este" en "este es el día. "
(?(_expresión_**)** _Sí_ **|**_no_**)**	Coincide con *sí* si el patrón de expresión regular designado por *expresión* coincide; de lo contrario, coincide con la parte opcional *no*. *expresión* se interpreta como una aserción de ancho cero.	`(?(A)A\d{2}\b|\b\d{3}\b)`	"A10", "910" en "A10 C103 910"
(?(_name_ **)** _Sí_ **|**_no_**)**	Coincide con *sí* si *nombre*, un grupo de captura con nombre o numerado, tiene una coincidencia; de lo contrario, coincide con la parte opcional *no*.	`(?<quoted>")?(?(quoted).+?"|\S+\s)`	Dogs.jpg, "Yiska playing.jpg" en "Dogs.jpg "Yiska playing.jpg""

Volver al principio

Sustituciones

Las sustituciones son elementos del lenguaje de expresiones regulares que se admiten en modelos de reemplazo. Para obtener más información, véase Novedades en el SDK Visual Studio 2015. Los metacaracteres enumerados en la tabla siguiente son aserciones atómicas de ancho cero.

Carácter	Descripción	Modelo	Modelo de reemplazo	Cadena de entrada	Cadena de resultado
$_número_	Sustituye la subcadena que coincide con el grupo *número*.	`\b(\w+)(\s)(\w+)\b`	`$3$2$1`	"one two"	"two one"
${ _name_ **}**	Sustituye la subcadena que coincide con el grupo con nombre *nombre*.	`\b(?<word1>\w+)(\s)(?<word2>\w+)\b`	`${word2}${word1}`	"one two"	"two one"
$$	Sustituye un "$" literal.	`\b(\d+)\s?USD`	`$$$1`	"103 USD"	"$103"
$&	Sustituye una copia de toda la coincidencia.	`\$?\d*\.?\d+`	`**$&**`	"$1.30"	"**$1.30**"
$`	Sustituye todo el texto de la cadena de entrada delante de la coincidencia.	`B+`	`$``	"AABBCC"	"AAAACC"
$'	Sustituye todo el texto de la cadena de entrada detrás de la coincidencia.	`B+`	`$'`	"AABBCC"	"AACCCC"
$+	Sustituye el último grupo capturado.	`B+(C+)`	`$+`	"AABBCCDD"	AACCDD
$_	Sustituye toda la cadena de entrada.	`B+`	`$_`	"AABBCC"	"AAAABBCCCC"

Volver al principio

Opciones de expresiones regulares

Puede especificar opciones que controlen cómo debe interpretar el motor de expresiones regulares un patrón de expresión regular. Muchas de estas opciones pueden especificarse alineadas (en el patrón de expresión regular) o como una o más constantes de RegexOptions. Esta referencia rápida solo muestra las opciones alineadas. Para obtener más información sobre las opciones alineadas y RegexOptions, consulte el artículo Opciones de expresiones regulares.

Puede especificar una opción alineada de dos formas:

- Con la construcción miscelánea **(?imnsx-imnsx)**, donde el signo menos (**-**) delante de una opción o un conjunto de opciones desactiva dichas opciones. Por ejemplo, **(?i-mn)** activa una coincidencia sin distinción entre mayúsculas y minúsculas (**i**), desactiva el modo multilínea (**m**) y desactiva las capturas de grupo sin nombre (**n**). La opción se aplica al patrón de expresión regular a partir del punto en el que esta se define y es efectiva hasta el final del patrón o hasta el punto en el que otro constructor invierte la opción.

- Con la construcción de agrupamiento **(?imnsx-imnsx:*subexpresión*)**, que define opciones solo para el grupo especificado.

El motor de expresiones regulares de .NET Framework admite las siguientes opciones alineadas.

Opción	Descripción	Modelo	Coincidencias
i	Usa la coincidencia sin distinción entre mayúsculas y minúsculas.	**\b(?i)a(?-i)a\w+\b**	"aardvark", "aaaAuto" en "aardvark AAAuto aaaAuto Adam breakfast"
m	Usa el modo multilínea. ^ y **$** coinciden con el principio y el final de una línea, en lugar del principio y el final de una cadena.	Para obtener un ejemplo, consulte la sección "Modo multilínea" en Opciones de expresiones regulares.	
n	No se capturan grupos sin nombre.	Para obtener un ejemplo, consulte la sección "Solo capturas explícitas" en Opciones de expresiones regulares.	
s	Usa el modo de una sola línea.	Para obtener un ejemplo, consulte la sección "Modo de una sola línea" en Opciones de expresiones regulares.	
x	Se omite el espacio en blanco sin escape en el patrón de expresión regular.	**\b(?x) \d+ \s \w+**	"1 aardvark", "2 cats" en "1 aardvark 2 cats IV centurions"

Volver al principio

Construcciones misceláneas

Las estructuras misceláneas modifican un modelo de expresión regular o proporcionan información sobre él. En la tabla siguiente se enumeran las construcciones misceláneas admitidas por .NET Framework. Para obtener más información, véase Novedades en el SDK Visual Studio 2015.

Construcción	Definición	Ejemplo
(?imnsx-imnsx)	Establece o deshabilita opciones como la no distinción entre mayúsculas y minúsculas en medio de un modelo. Para obtener más información, consulta Opciones de expresiones regulares.	`\bA(?i)b\w+\b` coincide con "ABA", "Able" en "ABA Able Act"
(?# *comentario*)	Comentario alineado. El comentario termina en el primer paréntesis de cierre.	`\bA(?#Matches words starting with A)\w+\b`
# [hasta el final de la línea]	Comentario en modo X. El comentario comienza en un carácter **#** sin escape y continúa hasta el final de la línea.	`(?x)\bA\w+\b#Matches words starting with A`

Ver también

System.Text.RegularExpressions
Regex
Expresiones regulares de .NET Framework
Clases de expresiones regulares
Ejemplos de expresiones regulares
Expresiones regulares: referencia rápida (descarga en formato Word)
Expresiones regulares: referencia rápida (descarga en formato PDF)