Sprachelemente für reguläre Ausdrücke – Kurzübersicht

Artikel • 12.12.2022 • 12 Minuten Lesedauer

Reguläre Ausdrücke sind Muster, für die die Engine für reguläre Ausdrücke eine Entsprechung im Eingabetext sucht. Muster können aus einem oder mehr Zeichenliteralen, Operatoren oder Konstrukten bestehen. Eine kurze Einführung finden Sie unter Reguläre Ausdrücke von .NET.

Jeder Abschnitt dieser Kurzübersicht enthält eine bestimmte Kategorie von Zeichen, Operatoren oder Konstrukten, mit denen Sie reguläre Ausdrücke definieren können.

Diese Informationen stehen in zwei weiteren Formaten zur Verfügung, die Sie zum Nachschlagen herunterladen und ausdrucken können:

- Download im Word-Format (DOCX)
- Download im PDF-Format (PDF)

Escapezeichen

Der umgekehrte Schrägstrich (\) in einem regulären Ausdruck gibt an, dass es sich bei dem darauf folgenden Zeichen um ein Sonderzeichen (wie in der folgenden Tabelle angezeigt) handelt oder dass das Zeichen als solches interpretiert werden soll. Weitere Informationen finden Sie unter Escapezeichen.

Escapezeichen	Beschreibung	Muster	Übereinstimmungen
`\a`	Entspricht einem Klingelzeichen (Warnsignal) \u0007.	`\a`	`"\u0007"` in `"Error!" + '\u0007'`
`\b`	Entspricht in einer Zeichenklasse einem Rücktastenzeichen \u0008.	`[\b]{3,}`	`"\b\b\b\b"` in `"\b\b\b\b"`
`\t`	Entspricht einem Tabstoppzeichen \u0009.	`(\w+)\t`	`"item1\t"`, `"item2\t"` in `"item1\titem2\t"`
`\r`	Entspricht einem Wagenrücklaufzeichen \u000D. (`\r` entspricht nicht dem Zeilenumbruchzeichen `\n`.)	`\r\n(\w+)`	`"\r\nThese"` in `"\r\nThese are\ntwo lines."`
`\v`	Entspricht einem vertikalen Tabstoppzeichen \u000B.	`[\v]{2,}`	`"\v\v\v"` in `"\v\v\v"`
`\f`	Entspricht einem Seitenvorschubzeichen \u000C.	`[\f]{2,}`	`"\f\f\f"` in `"\f\f\f"`
`\n`	Entspricht einer neuen Zeile \u000A.	`\r\n(\w+)`	`"\r\nThese"` in `"\r\nThese are\ntwo lines."`
`\e`	Entspricht einem Escapezeichen \u001B.	`\e`	`"\x001B"` in `"\x001B"`
`\` *Nnn*	Verwendet die oktale Darstellung, um ein Zeichen anzugeben (*nnn* besteht aus zwei oder drei Ziffern).	`\w\040\w`	`"a b"`, `"c d"` in `"a bc d"`

Escapezeichen	Beschreibung	Muster	Übereinstimmungen
`\x` *Nn*	Verwendet die hexadezimale Darstellung, um ein Zeichen anzugeben (*nn* besteht genau aus zwei Ziffern).	`\w\x20\w`	`"a b"`, `"c d"` in `"a bc d"`
`\c` *X* `\c` *x*	Entspricht dem durch *X* oder *x* angegebenen ASCII-Steuerzeichen, wobei *X* oder *x* der Buchstabe des Steuerzeichens ist.	`\cC`	`"\x0003"` in `"\x0003"` (STRG-C)
`\u` *Nnnn*	Entspricht einem Unicode-Zeichen in hexadezimaler Darstellung (genau vier Stellen, dargestellt durch *nnnn*).	`\w\u0020\w`	`"a b"`, `"c d"` in `"a bc d"`
`\`	Entspricht dem angegebenen Zeichen, wenn darauf ein Zeichen folgt, das in dieser und anderen Tabellen in diesem Thema nicht als Escapezeichen erkannt wird. Beispielsweise ist `\*` identisch mit `\x2A` und `\.` entspricht `\x2E`. Hierdurch kann die Engine für reguläre Ausdrücke Sprachelemente (z. B. * oder?) und Zeichenliterale (dargestellt durch `\*` oder `\?`) unterscheiden.	`\d+[\+-x\*]\d+`	`"2+2"` und `"3*9"` in `"(2+2) * 3*9"`

Zeichenklassen

Eine Zeichenklasse entspricht einer beliebigen Reihe von Zeichen. Zeichenklassen verwenden die in der folgenden Tabelle aufgeführten Sprachelemente. Weitere Informationen finden Sie unter Zeichenklassen in regulären Ausdrücken.

Zeichenklasse	Beschreibung	Muster	Übereinstimmungen

[*character_group*]	Entspricht jedem einzelnen Zeichen in *character_group*. Bei der Entsprechung wird standardmäßig die Groß- und Kleinschreibung berücksichtigt.	`[ae]`	`"a"` in `"gray"`
			
			`"a"`, `"e"` in `"lane"`

[^ *character_group*]	Negation: Entspricht einem einzelnen Zeichen, das sich nicht in *character_group* befindet. Standardmäßig wird bei Zeichen in *Zeichengruppe* die Groß-/Kleinschreibung beachtet.	`[^aei]`	`"r"`, `"g"`, `"n"` in `"reign"`
[*Ersten* - *letzte*]	Zeichenbereich: Entspricht einem einzelnen Zeichen im Bereich von *der ersten* bis *zum letzten*.	`[A-Z]`	`"A"`, `"B"` in `"AB123"`

.	Wildcard: Entspricht jedem einzelnen Zeichen außer \n.
	
	Um einem Literalperiodenzeichen (oder `\u002E`) zu entsprechen, müssen Sie ihm das Escapezeichen `\.` () vorangestellt haben.
		`a.e`	`"ave"` in `"nave"`
			
			`"ate"` in `"water"`

`\p{` *Namen* `}`	Entspricht einem einzelnen Zeichen in der allgemeinen Unicode-Kategorie oder einem benannten Block, der nach *Namen* angegeben wird.	`\p{Lu}`
		
		`\p{IsCyrillic}`
			`"C"`, `"L"` in `"City Lights"`
			
			`"Д"`, `"Ж"` in `"ДЖем"`

Zeichenklasse	Beschreibung	Muster	Übereinstimmungen
`\P{` *Namen* `}`	Entspricht einem einzelnen Zeichen, das sich nicht in der allgemeinen Unicode-Kategorie oder dem benannten Block befindet, der nach *Name* angegeben ist.	`\P{Lu}` `\P{IsCyrillic}`	`"i", "t", "y"` in `"City"` `"e", "m"` in `"ДЖem"`
`\w`	Entspricht einem beliebigen Wortzeichen.	`\w`	`"I", "D", "A", "1", "3"` in `"ID A1.3"`
`\W`	Entspricht jedem Nicht-Wort-Zeichen.	`\W`	`" ", "."` in `"ID A1.3"`
`\s`	Entspricht jedem Leerzeichen.	`\w\s`	`"D "` in `"ID A1.3"`
`\S`	Entspricht einem Nicht-Leerzeichen.	`\s\S`	`" _"` in `"int __ctr"`
`\d`	Entspricht einer dezimalen Ziffer.	`\d`	`"4"` in `"4 = IV"`
`\D`	Entspricht einem anderen Zeichen als einer Dezimalzahl.	`\D`	`" ", "=", " ", "I", "V"` in `"4 = IV"`

Anchors

Anchors oder atomare Assertionen mit einer Breite von Null bewirken, dass, in Abhängigkeit von der Position in der Zeichenfolge, eine Entsprechung gefunden oder nicht gefunden wird. Sie bewirken jedoch nicht, dass die Engine die Zeichenfolge durchläuft oder Zeichen verwendet. Die Metazeichen in der folgenden Tabelle sind Anchors. Weitere Informationen finden Sie unter Anchor.

Assertion	Beschreibung	Muster	Übereinstimmungen
`^`	Die Übereinstimmung muss standardmäßig zu Anfang der Zeichenfolge beginnen. Im Mehrzeilenmodus muss sie am Anfang der Zeile beginnen.	`^\d{3}`	`"901"` in `"901-333-"`
`$`	Die Übereinstimmung muss standardmäßig am Ende der Zeichenfolge oder vor `\n` am Ende der Zeichenfolge stattfinden. Im Mehrzeilenmodus muss sie am Ende der Zeile oder vor `\n` am Ende der Zeile erfolgen.	`-\d{3}$`	`"-333"` in `"-901-333"`
`\A`	Der Vergleich muss am Beginn der Zeichenfolge erfolgen.	`\A\d{3}`	`"901"` in `"901-333-"`
`\Z`	Der Vergleich muss am Ende der Zeichenfolge oder vor `\n` am Ende der Zeichenfolge erfolgen.	`-\d{3}\Z`	`"-333"` in `"-901-333"`
`\z`	Der Vergleich muss am Ende der Zeichenfolge erfolgen.	`-\d{3}\z`	`"-333"` in `"-901-333"`
`\G`	Die Übereinstimmung muss an dem Punkt auftreten, an dem die vorherige Übereinstimmung beendet wurde, oder wenn keine vorherige Übereinstimmung vorhanden war, an der Position in der Zeichenfolge, an der die Übereinstimmung gestartet wurde.	`\G\(\d\)`	`"(1)", "(3)", "(5)"` in `"(1)(3)(5)[7](9)"`
`\b`	Der Vergleich muss an einer Begrenzung zwischen einem `\w` (alphanumerischen) und einem `\W` (nicht alphanumerischen) Zeichen erfolgen.	`\b\w+\s\w+\b`	`"them theme", "them them"` in `"them theme them them"`

Assertion	Beschreibung	Muster	Übereinstimmungen
`\B`	Der Vergleich darf nicht an einer `\b`-Begrenzung erfolgen.	`\Bend\w*\b`	`"ends", "ender"` in `"end sends endure lender"`

Gruppierungskonstrukte

Gruppierungskonstrukte grenzen Teilausdrücke eines regulären Ausdrucks ab und zeichnen gewöhnlich Teilzeichenfolgen einer Eingabezeichenfolge auf. Gruppierungskonstrukte verwenden die Sprachelemente in der folgenden Tabelle. Weitere Informationen finden Sie unter Gruppierungskonstrukte.

Gruppierungskonstrukt	Beschreibung	Muster	Übereinstimmungen
`(` *Teilausdruck* `)`	Zeichnet den übereinstimmenden Teilausdruck auf und weist diesem eine einsbasierte Ordinalzahl zu.	`(\w)\1`	`"ee"` in `"deep"`
`(?<` *Namen* `>` *Teilausdruck* `)` oder `(?'` *Namen* `'` *Teilausdruck* `)`	Zeichnet den übereinstimmenden Teilausdruck in einer benannten Gruppe auf.	`(?<double>\w)\k<double>`	`"ee"` in `"deep"`
`(?<` *Name1* `-` *Name2* `>` *Teilausdruck* `)` oder `(?'` *Name1* `-` *Name2* `'` *Teilausdruck* `)`	Definiert eine Ausgleichsgruppendefinition. Weitere Informationen finden Sie im Abschnitt „Ausgleichen von Gruppendefinitionen“ in Gruppierungskonstrukte.	`(((?'Open'\()[^\(\)]*)+((?'Close-Open'\))[^\(\)]*)+)*(?(Open)(?!))$`	`"((1-3)*(3-1))"` in `"3+2^((1-3)*(3-1))"`

`(?:` *Teilausdruck* `)`	Definiert eine Nicht-Erfassungsgruppe.	`Write(?:Line)?`	`"WriteLine"` in `"Console.WriteLine()"`
			
			`"Write"` in `"Console.Write(value)"`

`(?imnsx-imnsx:` *Teilausdruck* `)`	Aktiviert oder deaktiviert die angegebenen Optionen in *Teilausdruck*. Weitere Informationen finden Sie unter Optionen für reguläre Ausdrücke.	`A\d{2}(?i:\w+)\b`	`"A12xl", "A12XL"` in `"A12xl A12XL a12xl"`
`(?=` *Teilausdruck* `)`	Positive Lookaheadassertion mit einer Breite von Null.	`\b\w+\b(?=.+and.+)`	`"cats", "dogs"` in `"cats, dogs and some mice."`
`(?!` *Teilausdruck* `)`	Negative Lookaheadassertion mit einer Breite von Null.	`\b\w+\b(?!.+and.+)`	`"and", "some", "mice"` in `"cats, dogs and some mice."`

Gruppierungskonstrukt	Beschreibung	Muster	Übereinstimmungen

`(?<=` *Teilausdruck* `)`	Positive Lookbehindassertion mit einer Breite von Null.	`\b\w+\b(?<=.+and.+)`
		———
		`\b\w+\b(?<=.+and.*)`
			`"some", "mice"` in `"cats, dogs and some mice."`
			———
			`"and", "some", "mice"` in `"cats, dogs and some mice."`

`(?<!` *Teilausdruck* `)`	Negative Lookbehindassertion mit einer Breite von Null.	`\b\w+\b(?<!.+and.+)`
		———
		`\b\w+\b(?<!.+and.*)`
			`"cats", "dogs", "and"` in `"cats, dogs and some mice."`
			———
			`"cats", "dogs"` in `"cats, dogs and some mice."`

`(?>` *Teilausdruck* `)`	Atomische Gruppe	`(?>a|ab)c`	`"ac"` In `"ac"` *Nichts* in `"abc"`

Schauen Sie sich auf einen Blick an

Wenn das reguläre Ausdrucksmodul auf einen **Lookaround-Ausdruck** trifft, dauert es eine Teilzeichenfolge, die von der aktuellen Position bis zum Start (Lookbehind) oder Ende (Lookahead) der ursprünglichen Zeichenfolge erreicht wird, und wird dann mithilfe des Lookaround-Musters auf dieser Unterzeichenfolge ausgeführt Regex.IsMatch . Der Erfolg dieses Subexpression-Ergebnisses wird dann bestimmt, ob es sich um eine positive oder negative Behauptung handelt.

Lookaround	Name	Funktion
`(?=check)`	Positives Lookahead	Gibt an, dass die aktuelle Position in der Zeichenfolge "überprüfen" unmittelbar folgt.
`(?<=check)`	Positiver Lookbehind	Stellt fest, dass die aktuelle Position in der Zeichenfolge "überprüfen" unmittelbar vor der aktuellen Position liegt.
`(?!check)`	Negatives Lookahead	Stellt fest, dass die aktuelle Position in der Zeichenfolge nicht "überprüft" ist, was sofort der aktuellen Position in der Zeichenfolge folgt.
`(?<!check)`	Negativer Lookbehind	Gibt an, dass die aktuelle Position in der Zeichenfolge nicht "überprüft" ist, was unmittelbar vor der aktuellen Position in der Zeichenfolge liegt.

Sobald sie übereinstimmen, werden **Atomgruppen** nicht erneut ausgewertet, auch wenn der Rest des Musters aufgrund der Übereinstimmung fehlschlägt. Dies kann die Leistung erheblich verbessern, wenn Quantifizierer innerhalb der Atomgruppe oder im Rest des Musters auftreten.

Quantifizierer

Quantifizierer geben an, wie viele Instanzen des vorherigen Elements (bei dem es sich um ein Zeichen

Quantifizierer geben an, wie viele Instanzen des vorherigen Elements (bei dem es sich um ein Zeichen, eine Gruppe oder eine Zeichenklasse handeln kann) in der Eingabezeichenfolge vorhanden sein müssen, damit eine Entsprechung gefunden wird. Quantifizierer verwenden die Sprachelemente in der folgenden Tabelle. Weitere Informationen finden Sie unter Quantifizierer in regulären Ausdrücken.

Quantifizierer	Beschreibung	Muster	Übereinstimmungen
`*`	Entspricht dem vorangehenden Element nicht oder mehrmals.	`a.*c`	`"abcbc"` in `"abcbc"`
`+`	Entspricht dem vorangehenden Element einmal oder mehrmals.	`"be+"`	`"bee"` in `"been"`, `"be"` in `"bent"`
`?`	Entspricht dem vorangehenden Element nicht oder einmal.	`"rai?"`	`"rai"` in `"rain"`
`{N}`	Entspricht dem vorangehenden Element genau *n*-mal.	`",\d{3}"`	`",043"` in `"1,043.6"`, `",876"`, `",543"` und `",210"` in `"9,876,543,210"`
`{N,}`	Entspricht dem vorangehenden Element mindestens *n*-mal.	`"\d{2,}"`	`"166"`, `"29"`, `"1930"`

`{N,M}`	Entspricht dem vorangehenden Element mindestens *n*-, höchstens jedoch *m*-mal.	`"\d{3,5}"`	`"166"`, `"17668"`
			
			`"19302"` in `"193024"`

`*?`	Entspricht dem vorangehenden Element nicht oder mehrmals, jedoch so wenige Male wie möglich.	`a.*?c`	`"abc"` in `"abcbc"`
`+?`	Entspricht dem vorangehenden Element ein- oder mehrmals, jedoch so wenige Male wie möglich.	`"be+?"`	`"be"` in `"been"`, `"be"` in `"bent"`
`??`	Entspricht dem vorangehenden Element nicht oder einmal, jedoch so wenige Male wie möglich.	`"rai??"`	`"ra"` in `"rain"`
`{N}?`	Entspricht dem vorangehenden Element genau *n*-mal.	`",\d{3}?"`	`",043"` in `"1,043.6"`, `",876"`, `",543"` und `",210"` in `"9,876,543,210"`
`{N,}?`	Entspricht dem vorangehenden Element mindestens *n*-mal, jedoch so wenige Male wie möglich.	`"\d{2,}?"`	`"166"`, `"29"`, `"1930"`

`{N,M}?`	Entspricht dem vorangehenden Element zwischen *n*- und *m*-mal, jedoch so wenige Male wie möglich.	`"\d{3,5}?"`	`"166"`, `"17668"`
			
			`"193"`, `"024"` in `"193024"`

Rückverweiskonstrukte

Ein Rückverweis ermöglicht es, einen zuvor gefundenen Teilausdruck später im gleichen regulären Ausdruck zu identifizieren. In der folgenden Tabelle sind die Rückverweiskonstrukte aufgeführt, die von regulären .NET-Ausdrücken unterstützt werden. Weitere Informationen finden Sie unter Rückverweiskonstrukte in regulären Ausdrücken.

Rückverweiskonstrukt	Beschreibung	Muster	Übereinstimmungen
`\` *Zahl*	Rückverweis. Entspricht dem Wert eines nummerierten Teilausdrucks.	`(\w)\1`	`"ee"` in `"seek"`
`\k<` *Namen* `>`	Benannter Rückverweis. Entspricht dem Wert eines benannten Ausdrucks.	`(?<char>\w)\k<char>`	`"ee"` in `"seek"`

Alternierungskonstrukte

Alternierungskonstrukte ändern einen regulären Ausdruck, um entweder/oder-Vergleiche zuzulassen. Diese Konstrukte verwenden die Sprachelemente in der folgenden Tabelle. Weitere Informationen finden Sie unter Alternierungskonstrukte in regulären Ausdrücken.

Alternierungskonstrukt	Beschreibung	Muster	Übereinstimmungen
`|`	Entspricht jedem beliebigen durch einen senkrechten Strich (|) getrennten Element.	`th(e|is|at)`	`"the"`, `"this"` in `"this is the day."`
`(?(` *Ausdruck* `)` *Ja* | *Nein* `)` oder `(?(` *Ausdruck* `)` *Ja* `)`	Entspricht *ja* , wenn das von *Ausdruck* angegebene Muster für reguläre Ausdrücke übereinstimmt. Andernfalls entspricht es dem optionalen *nein*. *Ausdruck* wird als Assertion mit einer Breite von Null interpretiert. Um die Unklarheit mit einer benannten oder nummerierten Aufnahmegruppe zu vermeiden, können Sie optional eine explizite Assertion wie folgt verwenden: `(?( (?=`*Ausdruck* `) )` *Ja* | *Nein* `)`	`(?(A)A\d{2}\b|\b\d{3}\b)`	`"A10"`, `"910"` in `"A10 C103 910"`
`(?(` *Namen* `)` *Ja* | *Nein* `)` oder `(?(` *Namen* `)` *Ja* `)`	Entspricht *ja*, wenn *Name*, eine benannte oder nummerierte Erfassungsgruppe, eine Übereinstimmung aufweist. Andernfalls entspricht es dem optionalen *nein*.	`(?<quoted>")?(?(quoted).+?"|\S+\s)`	`"Dogs.jpg "`, `"\"Yiska playing.jpg\""` in `"Dogs.jpg \"Yiska playing.jpg\""`

Ersetzungen

Ersetzungen sind Sprachelemente regulärer Ausdrücke, die in Ersetzungsmustern unterstützt werden. Weitere Informationen finden Sie unter Ersetzungen in regulären Ausdrücken. Die Metazeichen in der folgenden Tabelle sind atomare Assertionen mit einer Breite von Null.

Zeichen	Beschreibung	Muster	Ersetzungsmuster	Eingabezeichenfolge	Ergebniszeichenfolge
`$`*Zahl*	Ersetzt die untergeordnete Zeichenfolge, die der *Zahl* einer Gruppe entspricht.	`\b(\w+)(\s)(\w+)\b`	`$3$2$1`	`"one two"`	`"two one"`
`${`*Namen*`}`	Ersetzt die untergeordnete Zeichenfolge, die dem genannten *Namen* der Gruppe entspricht.	`\b(?<word1>\w+)(\s)(?<word2>\w+)\b`	`${word2} ${word1}`	`"one two"`	`"two one"`
`$$`	Ersetzt ein "$"-Literal.	`\b(\d+)\s?USD`	`$$$1`	`"103 USD"`	`"$103"`
`$&`	Ersetzt eine Kopie der gesamten Entsprechung.	`\$?\d*\.?\d+`	`**$&**`	`"$1.30"`	`"**$1.30**"`
`$``	Ersetzt den gesamten Text der Eingabezeichenfolge vor der Entsprechung.	`B+`	`$``	`"AABBCC"`	`"AAAACC"`
`$'`	Ersetzt den gesamten Text der Eingabezeichenfolge nach der Entsprechung.	`B+`	`$'`	`"AABBCC"`	`"AACCCC"`
`$+`	Ersetzt die zuletzt erfasste Gruppe.	`B+(C+)`	`$+`	`"AABBCCDD"`	`"AACCDD"`
`$_`	Ersetzt die gesamte Eingabezeichenfolge.	`B+`	`$_`	`"AABBCC"`	`"AAAABBCCCC"`

Optionen für reguläre Ausdrücke

Sie können Optionen angeben, die steuern, wie die Engine für reguläre Ausdrücke ein Muster des regulären Ausdrucks interpretiert. Viele dieser Optionen können entweder inline (im Muster des regulären Ausdrucks) oder als eine oder mehrere RegexOptions-Konstanten angegeben werden. Diese Kurzübersicht enthält nur Inlineoptionen. Weitere Informationen zu Inlineoptionen und RegexOptions-Optionen finden Sie im Artikel Optionen für reguläre Ausdrücke.

Sie können eine Inlineoption auf zwei Arten angeben:

- Wenn Sie das verschiedene Konstrukt `(?imnsx-imnsx)` verwenden, wobei ein Minuszeichen (-) vor einer Option oder einem Satz von Optionen diese Optionen deaktiviert. Zum Beispiel aktiviert `(?i-mn)` Übereinstimmungen ohne Berücksichtigung der Groß-/Kleinschreibung (`i`), deaktiviert Mehrzeilenmodus (`m`) und deaktiviert unbenannte Gruppenerfassungen (`n`). Die Option gilt für das

Muster des regulären Ausdrucks ab dem Punkt, an dem die Option definiert ist, und ist entweder

Muster des regulären Ausdrucks ab dem Punkt, an dem die Option definiert ist, und ist entweder bis zum Ende des Musters oder bis zu dem Punkt gültig, an dem ein anderes Konstrukt die Option umkehrt.

- Mit Gruppierungskonstrukte `(?imnsx-imnsx:` *Teilausdruck* `)`, das die Optionen nur für die angegebene Gruppe definiert.

Die .NET-Engine für reguläre Ausdrücke unterstützt die folgenden Inlineoptionen:

Option	Beschreibung	Muster	Übereinstimmungen
`i`	Groß-/Kleinschreibung bei der Suche ignorieren	`\b(?i)a(?-i)a\w+\b`	`"aardvark"`, `"aaaAuto"` in `"aardvark AAAuto aaaAuto Adam breakfast"`
`m`	Mehrzeilenmodus verwenden. `^` und `$` entsprechen dem Anfang und Ende einer Zeile anstatt dem Anfang und Ende einer Zeichenfolge.	Ein Beispiel finden Sie im Abschnitt zum Mehrzeilenmodus in Optionen für reguläre Ausdrücke.	
`n`	Unbenannte Gruppen nicht erfassen	Ein Beispiel finden Sie im Abschnitt zu ausschließlich expliziten Erfassungen in Optionen für reguläre Ausdrücke.	
`s`	Einzeilenmodus verwenden	Ein Beispiel finden Sie im Abschnitt zum Einzeilenmodus in Optionen für reguläre Ausdrücke.	
`x`	Leerraum ohne Escapezeichen im Muster eines regulären Ausdrucks ignorieren	`\b(?x) \d+ \s \w+`	`"1 aardvark"`, `"2 cats"` in `"1 aardvark 2 cats IV centurions"`

Verschiedene Konstrukte

Verschiedene Konstrukte ändern Muster von regulären Ausdrücken oder stellen Informationen darüber bereit. In der folgenden Tabelle sind die verschiedenen Konstrukte aufgeführt, die von .NET unterstützt werden. Weitere Informationen finden Sie unter Verschiedene Konstrukte.

Konstrukt	Definition	Beispiel
`(?imnsx-imnsx)`	Aktiviert oder deaktiviert Optionen wie die Groß-/Kleinschreibung mitten in einem Muster. Weitere Informationen finden Sie unter Optionen für reguläre Ausdrücke.	`\bA(?i)b\w+\b` entspricht `"ABA"`, `"Able"` in `"ABA Able Act"`
`(?#` *Kommentar* `)`	Inlinekommentar. Der Kommentar endet bei der ersten schließenden Klammer.	`\bA(?#Matches words starting with A)\w+\b`
`#` [bis Zeilenende]	X-Modus-Kommentar. Der Kommentar beginnt bei einem `#` ohne Escapezeichen und reicht bis zum Ende der Zeile.	`(?x)\bA\w+\b#Matches words starting with A`

Siehe auch

- System.Text.RegularExpressions
- System.Text.RegularExpressions.Regex
- Reguläre Ausdrücke von .NET
- Das Objektmodell für reguläre Ausdrücke
- Reguläre Ausdrücke – Kurzübersicht (Download im Word-Format)
- Reguläre Ausdrücke – Kurzübersicht (Download im PDF-Format)