Regex: I want this AND that AND that… in any order

Ask Question

38

21

I'm not even sure if this is possible or not, but here's what I'd like.

```
String: "NS306 FEBRUARY 20078/9/201013B1-9-1Lo
```

I have a text box where I type in the search parameters and they are space delimited. Because of this, I want to return a match is string1 is in the string and then string2 is in the string, OR string2 is in the string and then string1 is in the string. I don't care what order the strings are in, but they ALL (will somethings me more than 2) have to be in the string.

So for instance, in the provided string I would want:

```
"FEB Low"
```

or

```
"Low FEB"
```

...to return as a match.

I'm REALLY new to regex, only read some tutorials on here but that was a while ago and I need to get this done today. Monday I start a new project which is much more important and can't be distracted with this issue. Is there anyway to do this with regular expressions, or do I have to iterate through each part of the search filter and permutate the order? Any and all help is extremely appreciated. Thanks.

UPDATE: The reason I don't want to iterate through a loop and am looking for the best performance wise is because unfortunately, the dataTable I'm using calls this function on every key press, and I don't want it to bog down.

UPDATE: Thank you everyone for your help, it was much appreciated.

CODE UPDATE:

Ultimately, this is what I went with.

```
string sSearch = nvc["sSearch"].ToString().Rep
if (sSearch != null && sSearch != "")
{
  Regex r = new Regex("^(?=.*" + sSearch + ").
  _AdminList = _AdminList.Where<IPB>(
                                     delegat
                                     {
                                        //Co
                                        bool
                                        retu
                                  }).ToList<
                                     }
}
```

The IPB class has X number of elements and in no one table throughout the site I'm working on are the columns in the same order. Therefore, I needed to any order search and I didn't want to have to write a lot of code to do it. There were other good ideas in here, but I know my boss really likes Regex (preaches them) and therefore I thought it'd be best if I went with that for now. If for whatever reason the site's performance slips (intranet site) then I'll try another way. Thanks everyone.

c# regex

share
improve this question

edited Aug 25 '10 at 15:21

asked Aug 20 '10 at 17:38
XstreamINsanity
2,015 9 34 52

add a comment

7 Answers

active oldest votes

83

You can use `(?=…)` *positive lookahead*; it asserts that a given pattern can be matched. You'd anchor at the beginning of the string, and one by one, in any order, look for a match of each of your patterns.

It'll look something like this:

```
^(?=.*one)(?=.*two)(?=.*three).*$
```

This will match a string that contains `"one"`, `"two"`, `"three"`, in any order (as seen on rubular.com).

Depending on the context, you may want to anchor on `\A` and `\Z`, and use single-line mode so the dot matches everything.

This is not the most efficient solution to the problem. The best solution would be to parse out the words in your input and putting it into an efficient set representation, etc.

Related questions

- How does the regular expression `(?<=#)[^#]+(?=#)` work?

More practical example: password validation

Let's say that we want our password to:

- Contain between 8 and 15 characters
- Must contain an uppercase letter
- Must contain a lowercase letter
- Must contain a digit
- Must contain one of special symbols

Then we can write a regex like this:

```
^(?=.{8,15}$)(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])
 \__________/\_________/\_________/\_________/
    length      upper      lower      digit
```

share
improve this answer

edited May 23 '17 at 10:31
Community ♦
1 1

answered Aug 20 '10 at 17:54
polygenelubricants
271k 97 499 587

I'm not quite sure what you mean by "efficient set representation." Couldn't I do this: string strTest = "^(?=.*" + strSearch.Replace(" ", ")(?=.*") + ").*$" That would make everything into one string without having to iterate wouldn't it? – XstreamINsanity Aug 20 '10 at 18:06

Thanks, that worked pretty smoothly, didn't have to wait longer than 2 seconds on 1600+ records. – XstreamINsanity Aug 20 '10 at 18:16

@XstreamINsanity What pattern did you use for your test ? – Rusty Aug 20 '10 at 18:30

Will add code in just a minute. I have to manually type it out. – XstreamINsanity Aug 20 '10 at 18:36

It would be worth it to mention that * allows for ?=. to match after any amount of characters. It took me a little searching and testing to figure that out without being explicitly mentioned. – p1l0t Feb 12 '14 at 17:57

add a comment

4

Why not just do a simple check for the text since order doesn't matter?

```
string test = "NS306 FEBRUARY 20078/9/201013B1
test = test.ToUpper();
bool match = ((test.IndexOf("FEB") >= 0) && (t
```

Do you need it to use regex?

share
improve this answer

answered Aug 20 '10 at 17:58
Kelsey
38.8k 13 104 142

Because I was trying to avoid iteration if possible. I had heard the Regex engine was stronger than any foreach or for loop, so I wanted to use that. – XstreamINsanity Aug 20 '10 at 18:02

1 @XstreamINsanity regex is slooooow :) It's great for pattern matching complex patterns and making the matching process easier to code... but for speed not so much. – Kelsey Aug 20 '10 at 18:05

@Kelsey - Slower than a for loop? :) – XstreamINsanity Aug 20 '10 at 18:08

@XstreamINsanity There is no for loop in my example :P Internally, `IndexOf` probably has some type of iterating but internally so does Regex, except it has a whole bunch of language specific regex details to deal with as well. – Kelsey Aug 20 '10 at 18:10

Well, what I'm saying is that your example shows 2 statements of IndexOf, when I may have infinite number of statements, so I'd have to iterate somehow all of the posibilities, most likely with a for loop. – XstreamINsanity Aug 20 '10 at 18:14

show 2 more comments

3 I think the most expedient thing for today will be to `string.Split(' ')` the search terms and then iterate over the results confirming that `sourceString.Contains(searchTerm)`

```
var source = @"NS306 FEBRUARY 20078/9/201013B1
var search = "FEB Low";

var terms = search.Split(' ');

bool all_match = !terms.Any(term => !(source.C
```

Notice that we use `Any()` to set up a short-circuit, so if the first term fails to match, we skip checking the second, third, and so forth.

This is not a great use case for RegEx. The string manipulation necessary to take an arbitrary number of search strings and convert that into a pattern almost certainly negates the performance benefit of matching the pattern with the RegEx engine, though this may vary depending on what you're matching against.

You've indicated in some comments that you want to avoid a loop, but RegEx is not a one-pass solution. It is not hard to create horrifically non-performant searches that loop and step character by character, such as the infamous catastrophic backtracking, where a very simple match takes thousands of steps to return `false`.

share
improve this answer

edited Aug 20 '10 at 18:15

answered Aug 20 '10 at 17:52

Jay
43.1k 8 80 110

I figured a for loop in my code (for whatever reason, just thinking) would be more performance damage than whatever the regex engine does in the library. Yeah, it all gets chopped up into machine code, but I don't know how the regex is written, just that it was written by people smarter than me (I'm willing to admit it). :) – XstreamINsanity Aug 20 '10 at 18:18

@XstreamINsanity Yes, it is easy to make mistakes that forfeit performance, but it is always a question of "right tool for the job." I think the consensus here is that RegEx isn't it for this particular job. In the end, though, you may not see a difference, but it wouldn't be too hard to try it both ways for yourself and choose whatever works best, either (except for that looming deadline). – Jay Aug 20 '10 at 18:46

Exactly. Like I said in my last update, there were many great ideas and all of them could work probably, that's why I'm glad I can keep this question here and have the options available to me. I did try yours however it didn't seem to work. I could have missed something, but once I saw the Regex option I quit what I was doing. Thanks for your help. – XstreamINsanity Aug 20 '10 at 18:52

add a comment

2

```
var text = @"NS306Low FEBRUARY 2FEB0078/9/2010
var matches = Regex.Matches(text, @"(FEB)|(Low
foreach (Match match in matches)
{
   Console.WriteLine(match.Value);
}

Output:
Low
FEB
FEB
Low
```

Should get you started.

share
improve this answer

answered Aug 20 '10 at 17:55
Rusty
2,795 13 23

Yeah, I had that, but I wanted to avoid a for loop as much as possible, thinking it would hinder performance. – XstreamINsanity Aug 20 '10 at 18:02

@XstreamINsanity. The loop is just for printing them out. The matches collection has the results of the Regex, which only executes once. – Rusty Aug 20 '10 at 18:21

add a comment

2

The answer by @polygenelubricants is both complete and perfect but I had a case where I wanted to match a date and something else e.g. a 10-digit number so the lookahead does not match and I cannot do it with just lookaheads so I used named groups:

```
(?:.*(?P<1>[0-9]{10}).*(?P<2>2[0-9]{3}-(?:0?[0-9]|1[0-2])-(?:[0-2]?[0-9]|3[0-1])).*)+
```

and this way the number is always group 1 and the date is always group 2. Of course it has a few flaws but it was very useful for me and I just thought I should share it! (take a look https://www.debuggex.com/r/YULCcpn8XtysHfmE)

share
improve this answer

edited Nov 6 '16 at 19:01

answered Nov 6 '16 at 18:40
Panais
306 1 12

add a comment

0

You don't have to test each permutation, just split your search into multiple parts "FEB" and "Low" and make sure each part matches. That will be far easier than trying to come up with a regex which matches the whole thing in one go (which I'm sure is theoretically possible, but probably not practical in reality).

share
improve this answer

answered Aug 20 '10 at 17:46
Bryce Wagner
1,847 18 39

1 That may be one way I go about it. However, I'd really like to see if there's a Regex solution to it because it is my understanding, and I hope I'm correct and don't sound foolish, that the regex engine would be able to find it faster than me iterating throught he strings in the array. – XstreamINsanity Aug 20 '10 at 17:50

add a comment

0

Use string.Split(). It will return an array of subtrings thata re delimited by a specified string/char. The code will look something like this.

```
int maximumSize = 100;

string myString = "NS306 FEBRUARY
20078/9/201013B1-9-1Low31 AUGUST 19870";

string[] individualString = myString.Split('
', maximumSize);
```

For more informationhttp://msdn.microsoft.com/en-us/library/system.string.split.aspx

Edit: If you really wanted to use Regular Expressions this pattern will work. `[^ ]*` And you will just use Regex.Matches(); The code will be something like this:

```
string myString = "NS306 FEBRUARY 20078/9/201013B1-9-1Low31 AUGUST 19870";

string pattern = "[^ ]*";
Regex rgx = new Regex(pattern);

foreach(Match match in reg.Matches(s))

{

    //do stuff with match.value

}
```

share
improve this answer

edited Aug 20 '10 at 18:08

answered Aug 20 '10 at 17:55
Reflux
1,159 4 15 26

Splitting it is easy, I'm trying to not reiterate through a loop. – XstreamINsanity Aug 20 '10 at 18:06