Can I test if a regex is valid in C# without throwing exception

Asked 16 years, 2 months ago Modified 2 months ago

Viewed 52k times

66

I allow users to enter a regular expression to match IP addresses, for doing an IP filtration in a related system. I would like to validate if the entered regular expressions are valid as a lot of userse will mess op, with good intentions though.

I can of course do a Regex.IsMatch() inside a try/catch and see if it blows up that way, but are there any smarter ways of doing it? Speed is not an issue as such, I just prefer to avoid throwing exceptions for no reason.

c# regex

Share

Improve this question

Follow

edited Oct 20, 2008 at 14:48

Andy Lester
93.5k ● 15 ● 104 ● 159

asked Oct 20, 2008 at 14:41

Mark S. Rasmussen
35.4k ● 4 ● 42 ● 58

I have a method to test whether a RegEx is valid, but it just wraps the regex in a Try/Catch. I'm not sure if there's a better way to do this, but I couldn't find one. – Jon Tackabury Oct 20, 2008 at 14:46

do you mean blowing up on creating the actual Regex? new regex(str) ? – Nicholas Mancuso Oct 20, 2008 at 14:52

Allowing the users to enter a start and end value for each octet (or a similar solution) might be worth considering instead of regex. – Greg Oct 20, 2008 at 14:56

You might also consider using CIDR (192.168.0.0/24) if your IP address regex is for ranges. en.wikipedia.org/wiki/CIDR – Richard Szalay Nov 21, 2009 at 10:07

9 Answers

Sorted by: Highest score (default)

63

I think exceptions are OK in this case.

Just make sure to shortcircuit and eliminate the exceptions you can:

```
private static bool IsValidRegex(string pattern)
{
    if (string.IsNullOrWhiteSpace(pattern)) return fal

    try
    {
        Regex.Match("", pattern);
    }
    catch (ArgumentException)
    {
        return false;
    }
    return true;
}
```

Share Improve this answer

Follow

edited Oct 5, 2020 at 4:32

Shimmy Weitzhandler
105k ● 126 ● 434 ● 639

answered Nov 21, 2009 at 9:22

Jeff Atwood
63.9k ● 48 ● 151 ● 153

2 I wonder will JIT compiler be smart or dumb enough to optimize away the whole try catch block because the return value of a pure function is not used? – deerchao Nov 23, 2013 at 16:44

Would `IsMatch()` be any faster/better than `Match()`, seeing that we don't actually want to perform a match? Just like testing for primality is infinitely (well, almost) faster than actually finding the factors. – dotNET Aug 28, 2016 at 12:41

1 `IsMatch()` calls `internal Match Run(bool quick, int prevlen, string input, int beginning, int length, int startat)` with `quick` set to `true` while `Match()` calls it with `quick` set to `false`. It is indeed a bit faster, about **1-5%** according to my simple tests. – Mikael Dúi Bolinder Dec 29, 2017 at 9:14

4 How about just `new Regex(pattern)`? – Drew Noakes Mar 16, 2018 at 20:53

Question specifically asks if it can be done without handling an exception. – Bretton Wade Aug 12, 2019 at 18:35

43

As long as you catch very specific exceptions, just do the try/catch.

Exceptions are not evil if used correctly.

Share Improve this answer Follow

answered Oct 20, 2008 at 14:51

Robert Deml
12.5k ● 21 ● 69 ● 92

7 The question specifically asks if it can be done without handling an exception. – Bretton Wade Aug 12, 2019 at 18:34

5 `Exceptions are not evil if used correctly` but they are expensive to throw as they include a dump of the stack trace inside of them. Exceptions should be used for actual errors (ideally) and not for testing inputs – Liam Feb 3, 2021 at 9:29

1 `but they are expensive to throw` I just spent the day trying to convert MS `RegexParser.ScanRegex()` into something that will return just an `enum` representing the error. I would bet that while exceptions may be memory expensive to throw, that there would be a significant performance cost to adding a bunch of checks. if I finish it I'll benchmark it. – Michael Wagner Dec 12, 2021 at 23:19

1 I guess I'll eat my hat github.com/mwagnerEE/BenchmarkResults/blob/main/… – Michael Wagner Dec 13, 2021 at 3:16

8

Not without a lot of work. Regex parsing can be pretty involved, and there's nothing public in the Framework to validate an expression.

`System.Text.RegularExpressions.RegexNode.ScanRegex()` looks to be the main function responsible for parsing an expression, but it's internal (and throws exceptions for any invalid syntax anyway). So you'd be required to reimplement the parse functionality - which would undoubtedly fail on edge cases or Framework updates.

I think just catching the ArgumentException is as good an idea as you're likely to have in this situation.

Share Improve this answer Follow

edited Feb 3, 2021 at 9:29
Liam
29.4k ● 28 ● 137 ● 199

answered Oct 20, 2008 at 14:53
Mark Brackett
85.6k ● 17 ● 111 ● 155

1 We have TryParse to deal with potentially malformed numbers. We should have a TryRegex to do the same thing--return a failure rather than an exception. Debugging user-entered regexes is annoying! – Loren Pechtel Aug 16, 2016 at 20:42

I think its actually `System.Text.RegularExpressions.RegexParser.ScanRegex()` Source: [referencesource.microsoft.com/#System/regex/system/text/ …]() – Michael Wagner Dec 12, 2021 at 19:35

5

I've ever been use below function and have no problem with that. It uses exception and timeout both, but it's functional. Of course it works on .Net Framework >= 4.5.

```
public static bool IsValidRegexPattern(string pattern,
int maxSecondTimeOut = 20)
{
    if (string.IsNullOrEmpty(pattern)) return false;
    Regex re = new Regex(pattern, RegexOptions.None, n
maxSecondTimeOut));
    try { re.IsMatch(testText); }
    catch{ return false; } //ArgumentException or Rege
    return true;
}
```

Share Improve this answer

Follow

edited Oct 16 at 7:07

Shimmy Weitzhandler
105k ● 126 ● 434 ● 639

answered Jan 30, 2020 at 7:39

MiMFa
1,153 ● 13 ● 14

1 plz change the timespan, it should be TimeSpan(0,0,maxSecondsTime...) instead of hardcoded "20" – swe Mar 20, 2020 at 9:29

2 I changed it now. Thank you so much my friend. – MiMFa Mar 28, 2020 at 7:22

A malformed regex isn't the worst of reasons for an exception.

2

Unless you resign to a *very* limited subset of regex syntax - and then write a regex (or a parser) for that - I think you have no other way of testing if it is valid but to try to build a state machine from it and make it match something.

Share Improve this answer Follow

answered Oct 20, 2008 at 14:52

Tomalak
338k ● 68 ● 545 ● 635

2

Depending on who the target is for this, I'd be very careful. It's not hard to construct regexes that can backtrack on themselves and eat a lot of CPU and memory -- they can be an effective Denial of Service vector.

Share Improve this answer Follow

answered Oct 20, 2008 at 20:25

Clinton Pierce
13.1k ● 15 ● 64 ● 90

Is there no "stack overflow" protection in the .NET regex parsing library? Can you give me an example that might give me trouble? – Mark S. Rasmussen Oct 20, 2008 at 20:33

2 Regex.IsMatch("bbbbbbbbbb", "(.*){50}a"); – Hound May 9, 2012 at 11:46

2 With .NET 4.5, you can add a timeout value to your Regex object – Sparky Mar 30, 2014 at 14:18

0

In .NET, unless you write your own regular expression parser (which I would strongly advise against), you're almost certainly going to need to wrap the creation of the new Regex object with a try/catch.

Share Improve this answer Follow

answered Oct 20, 2008 at 14:56

theraccoonbear
4,327 ● 3 ● 35 ● 41

0

This is my solution, that outputs an enum telling whether the pattern is useable, and if yes, then return the compiled regex as an out parameter that you can use directly in your calling code. Regards.

```
namespace ProgrammingTools.Regex
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text.RegularExpressions;

    public enum eValidregex { No, Yes, YesButUseCompar

    public class RegEx_Validate
    {
        public static eValidregex IsValidRX ( string p
        {
            RX = null;

            if ( pattern.Length == 0 )
                return eValidregex.No;

            List<char> c1 = new List<char>
            {
                '\\' , '.' , '(' , ')' , '{' , '}' , '
'?' , '[' , ']', '|'
      };

      if ( c1.Count( e => pattern.Contains( e )
      {
          TimeSpan ts_timeout = new TimeSpan(day
0,seconds: 1,milliseconds: 0);

          try
          {
              RX = new Regex( pattern , RegexOpt
RegexOptions.IgnoreCase , ts_timeout );
              return eValidregex.Yes;
          }
          catch ( ArgumentNullException )
          {
              return eValidregex.No;
          }
          catch ( ArgumentOutOfRangeException )
          {
              return eValidregex.No;
          }
          catch ( ArgumentException )
          {
              return eValidregex.No;
          }
      }
      else
      {
          return eValidregex.YesButUseCompare;
      }

    }

  }

}
```

Share Improve this answer

Follow

answered Jul 6, 2022 at 13:19

Johan B

16 ● 1

-3

By using following method you can check wether your reguler expression is valid or not. here testPattern is the pattern you have to check.

```
public static bool VerifyRegEx(string testPattern)
{
    bool isValid = true;
    if ((testPattern != null) && (testPattern.Trim().L
    {
        try
        {
            Regex.Match("", testPattern);
        }
        catch (ArgumentException)
        {
            // BAD PATTERN: Syntax error
            isValid = false;
        }
    }
    else
    {
        //BAD PATTERN: Pattern is null or blank
        isValid = false;
    }
    return (isValid);
}
```

Share Improve this answer

Follow

edited Nov 22, 2013 at 8:15

Alex
23.3k ● 4 ● 42 ● 75

answered Nov 22, 2013 at 8:12

Niharika Singh
1

The title specifically says "without throwing exception"!
– rymdsmurf Oct 29, 2018 at 15:27

2 Trimming the pattern is also bad form since it might contain spaces for matching – mhapps Nov 11, 2019 at 9:33