Case insensitive 'Contains(string)'

Ask Question

2627

381

Is there a way to make the following return true?

```
string title = "ASTRINGTOTEST";
title.Contains("string");
```

There doesn't seem to be an overload that allows me to set the case sensitivity.. Currently I UPPERCASE them both, but that's just silly (by which I am referring to the i18n issues that come with up- and down casing).

UPDATE
This question is ancient and since then I have realized I asked for a simple answer for a really vast and difficult topic if you care to investigate it fully.
For most cases, in mono-lingual, English code bases this answer will suffice. I'm suspecting because most people coming here fall in this category this is the most popular answer.
This answer however brings up the inherent problem that we can't compare text case insensitive until we know both texts are the same culture and we know what that culture is. This is maybe a less popular answer, but I think it is more correct and that's why I marked it as such.

c# string contains case-insensitive

edited Jun 6 '18 at 14:52

Lonely Neuron
3,060 3 18 34

asked Jan 14 '09 at 21:39

Boris Callens
37k 69 196 289

24 Answers

1220

To test if the string `paragraph` contains the string `word` (thanks @QuarterMeister)

```
culture.CompareInfo.IndexOf(paragraph, word, CompareOptions.IgnoreC
```

Where `culture` is the instance of `CultureInfo` describing the language that the text is written in.

This solution is transparent about **the definition of case-insensitivity, which is language dependent**. For example, the English language uses the characters `I` and `i` for the upper and lower case versions of the ninth letter, whereas the Turkish language uses these characters for the eleventh and twelfth letters of its 29 letter-long alphabet. The Turkish upper case version of 'i' is the unfamiliar character 'İ'.

Thus the strings `tin` and `TIN` are the same word *in English*, but different words *in Turkish*. As I understand, one means 'spirit' and the other is an onomatopoeia word. (Turks, please correct me if I'm wrong, or suggest a better example)

To summarise, you can only answer the question 'are these two strings the same but in different cases' *if you know what language the text is in*. If you don't know, you'll have to take a punt. Given English's hegemony in software, you should probably resort to

`CultureInfo.InvariantCulture` , because it'll be wrong in familiar ways.

edited Aug 10 '17 at 12:06

alykhalid
1,176 6 18

answered Mar 17 '13 at 18:22

Colonel Panic
82.4k 62 305 396

58 Why not `culture.CompareInfo.IndexOf(paragraph, word, CompareOptions.IgnoreCase) >= 0` ? That uses the right culture and is case-insensitive, it doesn't allocate temporary lowercase strings, and it avoids the question of whether converting to lowercase and comparing is always the same as a case-insensitive comparison. – Quartermeister Mar 18 '13 at 15:32

9 This solution also needlessly pollutes the heap by allocating memory for what should be a searching function – JaredPar Mar 18 '13 at 16:09

15 Comparing with ToLower() will give different results from a case-insensitive IndexOf when two different letters have the same lowercase letter. For example, calling ToLower() on either U+0398 "Greek Capital Letter Theta" or U+03F4 "Greek Capital Letter Theta Symbol" results in U+03B8, "Greek Small Letter Theta", but the capital letters are considered different. Both solutions consider lowercase letters with the same capital letter different, such as U+0073 "Latin Small Letter S" and U+017F "Latin Small Letter Long S", so the IndexOf solution seems more consistent. – Quartermeister Mar 18 '13 at 17:47

17 +1 for completeness - answers with a proper form of explanation are the only way users will actually learn from SO – TheGeekZn May 5 '14 at 12:57

7 Why didn't you write "ddddfg".IndexOf("Df", StringComparison.OrdinalIgnoreCase) ? – Chen Aug 23 '15 at 13:41

0

Just to build on the answer here, you can create a string extension method to make this a littler user friendly:

```
    public static bool ContainsIgnoreCase(this string paragraph, str
    {
        return new CultureInfo("en-US").CompareInfo.IndexOf(paragrapl
CompareOptions.IgnoreCase) >= 0;
    }
```

answered Feb 14 at 8:53

Melbourne Developer
1,839 14 41

Assuming your paragraph and word will always be in en-US – Boris Callens Feb 15 at 13:42

127

Alternative solution using Regex:

```
bool contains = Regex.IsMatch("StRiNG to search", Regex.Escape("stri
RegexOptions.IgnoreCase);
```

edited Dec 17 '18 at 9:18

marsze
5,931 3 20 42

answered Jul 28 '10 at 17:18

Jed
6,724 17 69 115

5 Good Idea, also we have a lot of bitwise combinations in RegexOptions like `RegexOptions.IgnoreCase & RegexOptions.IgnorePatternWhitespace & RegexOptions.CultureInvariant;` for anyone if helps. – Saravanan Aug 24 '11 at 4:36

7 Must say I prefer this method although using IsMatch for neatness. – wonea Sep 7 '11 at 17:40

27 What's worse, since the search string is interpreted as a regex, a number of punctuation chars will cause incorrect results (or trigger an exception due to an invalid expression). Try searching for `"."` in `"This is a sample string that doesn't contain the search string"` . Or try searching for `"(invalid"` , for that matter. – cHao Sep 9 '11 at 13:28

13 @cHao: In that case, `Regex.Escape` could help. Regex still seems unnecessary when `IndexOf` / extension `Contains` is simple (and arguably more clear). – Dan Mangiarelli Sep 9 '11 at 16:44

5 Note that I was not implying that this Regex solution was the best way to go. I was simply adding to the list of answers to the original posted question "Is there a way to make the following return true?". – Jed Sep 13 '11 at 15:43

.NET Core 2.0+ only (as of now)

13

.NET Core has had a pair of methods to deal with this since version 2.0 :

- String.Contains(Char, **StringComparison**)
- String.Contains(String, **StringComparison**)

Example:

```
"Test".Contains("test", System.StringComparison.CurrentCultureIgnore(
```

In time, they will probably make their way into the .NET Standard and, from there, into all the other implementations of the Base Class Library.

edited Nov 8 '18 at 14:49
tronman

6,351 7 34 41

answered Oct 13 '18 at 9:26

Mathieu Renda

4,643 1 16 28

2 Finally ! It took a long time to be available... – AFract Feb 12 at 16:33

12

These are the easiest solutions.

1. By Index of

```
string title = "STRING";

if (title.IndexOf("string", 0, StringComparison.CurrentCultureIg
{
    // contains
}
```

2. By Changing case

```
string title = "STRING";

bool contains = title.ToLower().Contains("string")
```

3. By Regex

```
Regex.IsMatch(title, "string", RegexOptions.IgnoreCase);
```

answered Jul 12 '18 at 9:25

LAV VISHWAKARMA

749 8 21

Greate, thanks to first example!!! – Fox Feb 26 at 12:43

This is clean and simple.

35

```
Regex.IsMatch(file, fileNamestr, RegexOptions.IgnoreCase)
```

edited Jun 6 '18 at 14:54

Lonely Neuron
3,060 3 18 34

answered Nov 9 '12 at 4:25

takirala
1,311 1 12 29

24 This will match against a pattern, though. In your example, if `fileNamestr` has any special regex characters (e.g. `*`, `+`, `.`, etc.) then you will be in for quite a surprise. The only way to make this solution work like a proper `Contains` function is to escape `fileNamestr` by doing `Regex.Escape(fileNamestr)`. – Xåppll'-l0llwlg'l - Feb 3 '13 at 15:18

2510

You could use the String.IndexOf Method and pass `StringComparison.OrdinalIgnoreCase` as the type of search to use:

```
string title = "STRING";
bool contains = title.IndexOf("string", StringComparison.OrdinalIgr
```

Even better is defining a new extension method for string:

```
public static class StringExtensions
{
    public static bool Contains(this string source, string toCheck,
```

```
comp)
    {
        return source?.IndexOf(toCheck, comp) >= 0;
    }
}
```

Note, that null propagation `?.` is available since C# 6.0 (VS 2015), for older versions use

```
if (source == null) return false;
return source.IndexOf(toCheck, comp) >= 0;
```

USAGE:

```
string title = "STRING";
bool contains = title.Contains("string", StringComparison.OrdinalIg
```

edited May 8 '18 at 12:57

Vadim Ovchinnikov
7,398 4 28 51

answered Jan 14 '09 at 21:44

JaredPar
582k 121 1076 1354

2 Great string extension method! I've edited mine to check the source string is not null to prevent any object reference errors from occurring when performing .IndexOf(). – Richard Pursehouse Feb 8 '13 at 10:48

7 This gives the same answer as `paragraph.ToLower(culture).Contains(word.ToLower(culture))` with `CultureInfo.InvariantCulture` and it doesn't solve any localisation issues. Why over complicate things? stackoverflow.com/a/15464440/284795 – Colonel Panic Mar 17 '13 at 18:52

49 @ColonelPanic the `ToLower` version includes 2 allocations which are unnecessary in a comparison / search operation. Why needlessly

allocate in a scenario that doesn't require it? – JaredPar Mar 18 '13 at 16:09

4 @Seabiscuit that won't work because `string` is an `IEnumerable<char>` hence you can't use it to find substrings – JaredPar Nov 6 '14 at 17:55

4 A word of warning: The default for `string.IndexOf(string)` is to use the current culture, while the default for `string.Contains(string)` is to use the ordinal comparer. As we know, the former can be changed be picking a longer overload, while the latter cannot be changed. A consequence of this inconsistency is the following code sample:

```
Thread.CurrentThread.CurrentCulture =
CultureInfo.InvariantCulture; string self = "Waldstrasse";
string value = "straße";
Console.WriteLine(self.Contains(value));/* False */
Console.WriteLine(self.IndexOf(value) >= 0);/* True */
```

– Jeppe Stig Nielsen Feb 18 '16 at 9:40

-4

Simple way for newbie:

```
title.ToLower().Contains("string");//of course "string" is lowercase
```

edited Mar 23 '18 at 3:33

answered Apr 17 '17 at 8:04

O Thạnh Ldt

81 8

1 No this does not completely solve the problem. – Ketan Dubey Jun 23 '17 at 12:49

Downvote for just being incorrect. What if title = StRiNg? StRiNg != string and StRiNg != STRING – berniefitz Oct 21 '17 at 4:30

I was wrong. Edit answer as follows, too simple simple:
title.ToLower().Contains("string") // of course "string" is lowercase – O Thạnh Ldt Mar 23 '18 at 3:28

0

if you want to check if your passed string is in string then there is a simple method for that.

```
string yourStringForCheck= "abc";
string stringInWhichWeCheck= "Test abc abc";

bool isContaines = stringInWhichWeCheck.ToLower().IndexOf(yourStringI
> -1;

This boolean value will return if string contains or not
```

answered Nov 16 '17 at 12:23

shaishav shukla
168 3 10

2

The trick here is to look for the string, ignoring case, but to keep it exactly the same (with the same case).

```
var s="Factory Reset";
var txt="reset";
int first = s.IndexOf(txt, StringComparison.InvariantCultureIgnoreC
var subString = s.Substring(first - txt.Length, txt.Length);
```

Output is "Reset"

edited Oct 31 '17 at 15:16

Massimiliano Kraus
2,399 4 16 33

answered May 3 '16 at 14:36

Mr.B

1,811 10 24

0

```
public static class StringExtension
{
    #region Public Methods

    public static bool ExContains(this string fullText, string value
    {
        return ExIndexOf(fullText, value) > -1;
    }

    public static bool ExEquals(this string text, string textToCompa
    {
        return text.Equals(textToCompare, StringComparison.OrdinalIg
    }

    public static bool ExHasAllEquals(this string text, params strin
    {
        for (int index = 0; index < textArgs.Length; index++)
            if (ExEquals(text, textArgs[index]) == false) return fal
        return true;
    }

    public static bool ExHasEquals(this string text, params string[]
    {
        for (int index = 0; index < textArgs.Length; index++)
            if (ExEquals(text, textArgs[index])) return true;
        return false;
    }

    public static bool ExHasNoEquals(this string text, params string
    {
        return ExHasEquals(text, textArgs) == false;
    }

    public static bool ExHasNotAllEquals(this string text, params st
    {
        for (int index = 0; index < textArgs.Length; index++)
```

```
                if (ExEquals(text, textArgs[index])) return false;
            return true;
        }

        /// <summary>
        /// Reports the zero-based index of the first occurrence of the s
        /// in the current System.String object using
    StringComparison.InvariantCultureIgnoreCase.
        /// A parameter specifies the type of search to use for the spec
        /// </summary>
        /// <param name="fullText">
        /// The string to search inside.
        /// </param>
        /// <param name="value">
        /// The string to seek.
        /// </param>
        /// <returns>
        /// The index position of the value parameter if that string is f
        /// is not. If value is System.String.Empty, the return value is
        /// </returns>
        /// <exception cref="ArgumentNullException">
        /// fullText or value is null.
        /// </exception>
        public static int ExIndexOf(this string fullText, string value)
        {
            return fullText.IndexOf(value, StringComparison.OrdinalIgnor
        }

        public static bool ExNotEquals(this string text, string textToCo
        {
            return ExEquals(text, textToCompare) == false;
        }

        #endregion Public Methods
    }
```

answered Jun 14 '17 at 3:16

Final Heaven

41 6

You can use `string.indexof ()` function. This will be case insensitive

2

edited Dec 11 '16 at 13:41

FelixSFD

4,425 7 32 91

answered Dec 11 '16 at 13:39

Okan SARICA

77 1 11

2

```
if ("strcmpstring1".IndexOf(Convert.ToString("strcmpstring2"),
StringComparison.CurrentCultureIgnoreCase) >= 0){return true;}else{r
```

answered Oct 26 '16 at 14:34

Tamilselvan K

490 4 9

10

Just like this:

```
string s="AbcdEf";
if(s.ToLower().Contains("def"))
{
    Console.WriteLine("yes");
}
```

edited Jul 19 '16 at 19:23

johnnyRose

4,230 11 36 53

answered Jul 13 '14 at 9:54

cdytoby

502 6 22

2 This is not culture-specific and may fail for some cases. culture.CompareInfo.IndexOf(paragraph, word, CompareOptions.IgnoreCase) should be used. – hikalkan Jul 22 '14 at 7:50

2 Why avoid string.ToLower() when doing case-insensitive string comparisons? TI;Dr *It's costly because a new string is "manufactured".* – Liam Oct 10 '16 at 10:00

Using a RegEx is a straight way to do this:

6

```
Regex.IsMatch(title, "string", RegexOptions.IgnoreCase);
```

edited Jul 19 '16 at 19:23

johnnyRose
4,230 11 36 53

answered Sep 18 '13 at 13:08

Stend
111 1 4

4 Your answer is exactly the same as guptat59's but, as was pointed out on his answer, this will match a regular expression, so if the string you're testing contains any special regex characters it will not yield the desired result. – Casey Dec 9 '13 at 22:55

2 This is a straight up copy of this answer and suffers from the same issues as noted in that answer – Liam Oct 10 '16 at 10:04

Downvote for the same reason @Liam gave – berniefitz Oct 21 '17 at 4:28

Agreed. Study regular expressions – Ja Austin Dec 26 '17 at 5:14

5

This is quite similar to other example here, but I've decided to simplify enum to bool, primary because other alternatives are normally not needed. Here is my example:

```
public static class StringExtensions
{
    public static bool Contains(this string source, string toCheck, b
bCaseInsensitive )
    {
        return source.IndexOf(toCheck, bCaseInsensitive ?
StringComparison.OrdinalIgnoreCase : StringComparison.Ordinal) >= 0;
    }
}
```

And usage is something like:

```
if( "main String substring".Contains("SUBSTRING", true) )
....
```

answered Oct 17 '15 at 7:46

TarmoPikaro

1,892 1 17 31

215

You can use `IndexOf()` like this:

```
string title = "STRING";

if (title.IndexOf("string", 0, StringComparison.CurrentCultureIgnore(
{
    // The string exists in the original
}
```

Since 0 (zero) can be an index, you check against -1.

MSDN

> The zero-based index position of value if that string is found, or -1 if it is not. If value is String.Empty, the return value is 0.

edited Sep 15 '15 at 15:45

Liam
16.5k 16 78 131

answered Jan 14 '09 at 21:48

mkchandler
3,810 3 17 24

24

OrdinalIgnoreCase, CurrentCultureIgnoreCase or InvariantCultureIgnoreCase?

Since this is missing, here are some recommendations about when to use which one:

Dos

- Use `StringComparison.OrdinalIgnoreCase` for comparisons as your safe default for culture-agnostic string matching.
- Use `StringComparison.OrdinalIgnoreCase` comparisons for increased speed.
- Use `StringComparison.CurrentCulture-based` string operations when displaying the output to the user.
- Switch current use of string operations based on the invariant culture to use the non-linguistic `StringComparison.Ordinal` or `StringComparison.OrdinalIgnoreCase` when the comparison is linguistically irrelevant (symbolic, for example).
- Use `ToUpperInvariant` rather than `ToLowerInvariant` when normalizing strings for comparison.

Don'ts

- Use overloads for string operations that don't explicitly or implicitly specify the string comparison mechanism.
- Use `StringComparison.InvariantCulture` -based string operations in most cases; one of the few exceptions would be persisting linguistically meaningful but culturally-agnostic data.

Based on these rules you should use:

```
string title = "STRING";
if (title.IndexOf("string", 0, StringComparison.[YourDecision]) != -1
{
    // The string exists in the original
}
```

whereas [YourDecision] depends on the recommendations from above.

link of source: http://msdn.microsoft.com/en-us/library/ms973919.aspx

edited Jul 23 '15 at 10:30

answered Jun 17 '14 at 10:31

Fabian Bigler
6,684 2 25 50

what if you know you're always gonna get an english string. which one to use? – BKSpurgeon Mar 23 '17 at 23:43

1 @BKSpurgeon I'd use OrdinalIgnoreCase, if case does not matter – Fabian Bigler Mar 24 '17 at 8:09

49

One issue with the answer is that it will throw an exception if a string is null. You can add that as a check so it won't:

```
public static bool Contains(this string source, string toCheck, Stri
{
    if (string.IsNullOrEmpty(toCheck) || string.IsNullOrEmpty(source
        return true;

    return source.IndexOf(toCheck, comp) >= 0;
}
```

edited Jun 5 '15 at 6:02

fubo
31k 9 70 107

answered Dec 7 '10 at 21:11

FeiBao 飞豹
679 5 6

8 If toCheck is the empty string it needs to return true per the Contains documentation: "true if the value parameter occurs within this string, or if value is the empty string (""); otherwise, false." – amurra Feb 16 '11 at 16:13

3 Based on amurra's comment above, doesn't the suggested code need to be corrected? And shouldn't this be added to the accepted answer, so that the best response is first? – David White Aug 30 '11 at 3:43

12 Now this will return true if source is an empty string or null no matter what toCheck is. That cannot be correct. Also IndexOf already returns true if toCheck is an empty string and source is not null. What is needed here is a check for null. I suggest if (source == null || value == null) return false; – Colin Jul 1 '13 at 12:21

2 The source cant be null – Lucas Dec 14 '16 at 16:55

`if (string.IsNullOrEmpty(source)) return string.IsNullOrEmpty(toCheck);` – Kyle Delaney Apr 4 '18 at 13:39

10

The `InStr` method from the VisualBasic assembly is the best if you have a concern about internationalization (or you could reimplement it). Looking at in it dotNeetPeek shows that not only does it account for caps and lowercase, but also for kana type and full- vs. half-width characters (mostly relevant for Asian languages, although there are full-width versions of the Roman alphabet too). I'm skipping over some details, but check out the private method `InternalInStrText`:

```
private static int InternalInStrText(int lStartPos, string sSrc, str
{
  int num = sSrc == null ? 0 : sSrc.Length;
  if (lStartPos > num || num == 0)
    return -1;
  if (sFind == null || sFind.Length == 0)
    return lStartPos;
  else
    return Utils.GetCultureInfo().CompareInfo.IndexOf(sSrc, sFind, l
CompareOptions.IgnoreCase | CompareOptions.IgnoreKanaType | CompareO
}
```

answered Dec 6 '13 at 14:11

Casey
2,516 18 35

9

Use this:

```
string.Compare("string", "STRING", new System.Globalization.CultureI
System.Globalization.CompareOptions.IgnoreCase);
```

edited Jul 8 '13 at 8:10

Peter Mortensen
13.9k 19 87 113

answered Jul 11 '11 at 7:53

mr.martan
189 2 1

25 The questioner is looking for `Contains` not `Compare`. – DuckMaestro Jul 11 '11 at 8:05

@DuckMaestro, the accepted answer is implementing `Contains` with `IndexOf`. So this approach is equally helpful! The C# code example on this page is using string.Compare(). SharePoint team's choice that is! – vulcan raven Jan 5 '13 at 10:07

10

I know that this is not the C#, but in the framework (VB.NET) there is already such a function

```
Dim str As String = "UPPERlower"
Dim b As Boolean = InStr(str, "UpperLower")
```

C# variant:

```
string myString = "Hello World";
bool contains = Microsoft.VisualBasic.Strings.InStr(myString, "world
```

edited Sep 9 '11 at 13:55

answered Sep 9 '11 at 13:23

serhio
15.9k 40 183 333

Do you also know how it works internally? – Boris Callens Mar 18 '13 at 8:12

35

StringExtension class is the way forward, I've combined a couple of the posts above to give a complete code example:

```
public static class StringExtensions
{
    /// <summary>
    /// Allows case insensitive checks
    /// </summary>
    public static bool Contains(this string source, string toCheck, 
comp)
    {
        return source.IndexOf(toCheck, comp) >= 0;
    }
}
```

edited Jan 25 '11 at 6:58

abatishchev
70.5k 70 267 397

answered Nov 18 '10 at 16:48

Andrew
7,634 10 56 98

70

You could always just up or downcase the strings first.

```
string title = "string":
title.ToUpper().Contains("STRING")  // returns true
```

Oops, just saw that last bit. A case insensitive compare would * probably * do the same anyway, and if performance is not an issue, I don't see a problem with creating uppercase copies and

comparing those. I could have sworn that I once saw a case-insensitive compare once...

edited Jan 14 '09 at 21:54

answered Jan 14 '09 at 21:42

Ed S.
103k 18 150 224

3 Interestingly, I've seen ToUpper() recommended over the use of ToLower() in this sort of scenario, because apparently ToLower() can "lose fidelity" in certain cultures - that is, two different upper-case characters translate to the same lower-case character. – Matt Hamilton Jan 14 '09 at 21:47

106 Search for "Turkey test" :) – Jon Skeet Jan 14 '09 at 21:48

5 In some French locales, uppercase letters don't have the diacritics, so ToUpper() may not be any better than ToLower(). I'd say use the proper tools if they're available - case-insensitive compare. – Blair Conrad Jan 14 '09 at 22:03

4 Don't use ToUpper or ToLower, and do what Jon Skeet said – Peter Gfader Aug 21 '09 at 2:49

12 Just saw this again after two years and a new downvote... anyway, I agree that there are better ways to compare strings. However, not all programs will be localized (most won't) and many are internal or throwaway apps. Since I can hardly expect credit for advice best left for throwaway apps... I'm moving on :D – Ed S. Jan 25 '11 at 7:28

protected by Shankar Damodaran Jan 15 '14 at 17:55

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?