

Case insensitive 'Contains(string)'

Asked 10 years, 9 months ago Active 16 days ago Viewed 815k times

Is there a way to make the following return true?

2762

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



404

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

3,141 3 18 35

asked Jan 14 '09 at 21:39



Boris Callens

38.7k 69 199 292

24 Answers

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

1298

```
culture.CompareInfo.IndexOf(paragraph, word, CompareOptions.IgnoreCase) >= 0
```



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

1,197 6 18

answered Mar 17 '13 at 18:22



Colonel Panic

87.5k 66 318 406

-
- 63 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
-
- 6 Yes, you are very right about Turkish. `ı => I, i => İ` – [adyusuf](#) Apr 9 '13 at 13:58
-
- 8 Why didn't you write `"dddfg".IndexOf("Df", StringComparison.OrdinalIgnoreCase)` ? – [Chen](#) Aug 23 '15 at 13:41
-

You could use the [String.IndexOf Method](#) and pass [StringComparison.OrdinalIgnoreCase](#) as the type of search to use:

2605

```
string title = "STRING";
bool contains = title.IndexOf("string", StringComparison.OrdinalIgnoreCase) >= 0;
```

Even better is defining a new extension method for string:

```
public static class StringExtensions
{
    public static bool Contains(this string source, string toCheck, StringComparison
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.OrdinalIgnoreCase);
```

edited May 8 '18 at 12:57



Vadim Ovchinnikov

8,323 4 32 57

answered Jan 14 '09 at 21:44



JaredPar

608k 127 1116
1372

-
- 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
-
- 8 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 ✎
-
- 56 @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
-
- 5 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 by 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

You can use `IndexOf()` like this:

219

```
string title = "STRING";

if (title.IndexOf("string", 0, StringComparison.CurrentCultureIgnoreCase) != -1)
{
    // 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

17.4k 16 80 135

answered Jan 14 '09 at 21:48



mkchandler

3,869 3 17 24

Alternative solution using Regex:

137

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

edited Dec 17 '18 at 9:18



marsze

6,284 3 22 44

answered Jul 28 '10 at 17:18



Jed

6,944 17 71 117

6 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

31 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

- 17 @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
- 6 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

You could always just up or downcase the strings first.

77

```
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.](#)

106k 19 158 233

- 117 Search for "Turkey test" :) – [Jon Skeet](#) Jan 14 '09 at 21:48

- 6 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
- 5 Don't use `ToUpper` or `ToLower`, and do what Jon Skeet said – [Peter Gfader](#) Aug 21 '09 at 2:49
- 13 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
- 6 Is searching for "Turkey test" the same as searching for "TURKEY TEST"? – [JackAce](#) Jun 1 '18 at 16:24

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:

51

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

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

edited Jun 5 '15 at 6:02



fubo

32.7k

11

74

111

answered Dec 7 '10 at 21:11



FeiBao 飞豹

707

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
- 1 if (string.IsNullOrEmpty(source)) return string.IsNullOrEmpty(toCheck); – [Kyle Delaney](#) Apr 4 '18 at 13:39

This is clean and simple.

37

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

edited Jun 6 '18 at 14:54



Lonely Neuron

3,141

3

18

35

answered Nov 9 '12 at 4:25



takirala

1,419

1

13

30

- 28 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) . – [Xâppl'-'l0llwlg'l](#) - Feb 3 '13 at 15:18

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

35

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

edited Jan 25 '11 at 6:58



[abatishchev](#)

72.7k 70 270 406

answered Nov 18 '10 at 16:48



[Andrew](#)

7,757 10 56 98

why are you allowing ANOTHER layer of abstraction over StringComparison ? – |----- Jun 20 at 18:37

.NET Core 2.0+ only (as of now)

32

.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.CurrentCultureIgnoreCase);
```

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](#)

answered Oct 13 '18 at 9:26

[Mathieu Renda](#)



6,631

7

35

43



5,816

1

18

28

5 Finally ! It took a long time to be available... – [AFract](#) Feb 12 at 16:33



27



OrdinalIgnoreCase, CurrentCultureIgnoreCase or InvariantCultureIgnoreCase?

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

Do's

- 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](#)

7,153 3 28 53

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

These are the easiest solutions.

13

1. By Index of

```
string title = "STRING";

if (title.IndexOf("string", 0, StringComparison.CurrentCultureIgnoreCase) != -1)
{
    // 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](#)

819 9 21

Just like this:

11

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

edited Jul 19 '16 at 19:23



johnnyRose

4,439 11 37 54

answered Jul 13 '14 at 9:54



cdytoby

517 6 22

-
- 3 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
-
- 3 [Why avoid string.ToLower\(\) when doing case-insensitive string comparisons?](#) TL;Dr *It's costly because a new string is "manufactured"*. – Liam Oct 10 '16 at 10:00
-

11

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 dotNetPeek 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, string sFind)
{
    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, lStartPos,
            CompareOptions.IgnoreCase | CompareOptions.IgnoreKanaType | CompareOptions.IgnoreWidth);
}
```

answered Dec 6 '13 at 14:11



Casey

2,606 19 37

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

11

```
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](#)

16.4k

40

190

343

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

Use this:

10

```
string.Compare("string", "STRING", new System.Globalization.CultureInfo("en-US"),  
System.Globalization.CompareOptions.IgnoreCase);
```

edited Jul 8 '13 at 8:10

answered Jul 11 '11 at 7:53



[Peter Mortensen](#)

14.6k

19

89

118



[mr.martan](#)

199

2

2

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

Using a RegEx is a straight way to do this:

7

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

edited Jul 19 '16 at 19:23



johnnyRose

4,439 11 37 54

answered Sep 18 '13 at 13:08



Stend

121 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 – Jared Dec 26 '17 at 5:14
-

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:

5

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

2,290 1 20 35

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

3

```
public static bool ContainsIgnoreCase(this string paragraph, string word)
{
    return CultureInfo.CurrentCulture.CompareInfo.IndexOf(paragraph, word,
CompareOptions.IgnoreCase) >= 0;
}
```

edited Sep 22 at 4:42

answered Feb 14 at 8:53



Melbourne Developer

2,176 16 49

-
- 1 Assuming your paragraph and word will always be in en-US – [Boris Callens](#) Feb 15 at 13:42 ✎
-
- 3 To avoid issues with forcing the culture to en-US, use `return CultureInfo.CurrentCulture.CompareInfo.IndexOf(paragraph, word, CompareOptions.IgnoreCase) >= 0;` instead. – [AndrewWhalan](#) Jun 13 at 21:42
-

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

2

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

bool isContained = stringInWhichWeCheck.ToLower().IndexOf(yourStringForCheck.ToLower())
> -1;
```

This boolean value will return if the string is contained or not

edited Oct 12 at 8:00

answered Nov 16 '17 at 12:23



gusmally

72 1 14



shaishav shukla

238 4 11

2

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

answered Oct 26 '16 at 14:34



Tamilselvan K

570 5 9

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

2

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

Output is "Reset"

edited Oct 31 '17 at 15:16



Massimiliano Kraus

2,559 4 17 34

answered May 3 '16 at 14:36



Mr.B

2,018 12 29

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

2

edited Dec 11 '16 at 13:41



FelixSFD

4,664 7 32 97

answered Dec 11 '16 at 13:39



Okan SARICA

84 1 11

```
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 textToCompare)
    {
        return text.Equals(textToCompare, StringComparison.OrdinalIgnoreCase);
    }

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

```

        if (Equals(text, textArgs[index]) == false) return false;
    }
    return true;
}

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

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

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

/// <summary>
/// Reports the zero-based index of the first occurrence of the specified string
/// in the current System.String object using
StringComparison.InvariantCultureIgnoreCase.
/// A parameter specifies the type of search to use for the specified string.
/// </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 found, or -1 if it
/// is not. If value is System.String.Empty, the return value is 0.
/// </returns>
/// <exception cref="ArgumentNullException">
/// fullText or value is null.
/// </exception>
public static int ExIndexOf(this string fullText, string value)
{
    return fullText.IndexOf(value, StringComparison.OrdinalIgnoreCase);
}

public static bool ExNotEquals(this string text, string textToCompare)
{

```

```
        return ExEquals(text, textToCompare) == false;
    }

    #endregion Public Methods
}
```

answered Jun 14 '17 at 3:16



Final Heaven

41 6

 Simple way for newbie: -3

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



edited Mar 23 '18 at 3:33

answered Apr 17 '17 at 8:04



O Thanh Ldt

134 8

1 No this does not completely solve the problem. – [Ketan Dubey](#) Jun 23 '17 at 12:49Downvote 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 Thanh Ldt](#) Mar 23 '18 at 3:28 **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?

