# Why is there no Char.Empty like String.Empty?

Ask Question

232

18

Is there a reason for this? I am asking because if you needed to use lots of empty chars then you get into the same situation as you would when you use lots of empty strings.

Edit: The reason for this usage was this:

```
myString.Replace ('c', '')
```

So remove all instances of 'c's from myString.

c# .net string char bcl

edited Dec 29 '17 at 22:38
JohnOsborne
486 7 22

asked Sep 8 '10 at 17:44
Joan Venge
106k 176 404 627

1 Yeah I used that word for lack of a better word. i.e. the recommended way of using String.Empty instead of "". – Joan Venge Sep 8 '10 at 17:51

Thanks, do you know why it's not recommended anymore? Is it because of the compiler does it for you? – Joan Venge Sep 8 '10 at 18:02

Home

PUBLIC

**Stack Overflow**

Tags

Users

Jobs

**Teams**
Q&A for work

Learn More

`params char[] toRemove)` ? The intent will be clearly communicated and you will not risk mistyping anything. – bzlm Oct 5 '10 at 11:38

11 @Henk - The only reason I use string.Empty is because I find the null object provided by Empty expresses intent better than empty quotes. Empty quotes could result from a merge problem, or a bungled thought, or it could be the actual intent of that code, whereas Empty explicitly tells me that the developer intended for that string not to have data. – Ritch Melton May 21 '11 at 0:33

3 There is a difference between "" and the string.Empty. Not that anyone care, really, but "" creates an object, whereas string.Empty makes use of one already made. But again, it is so small, that only special situations it would make a diference – marcelo-ferraz Nov 30 '11 at 12:35

## 19 Answers

242 ✓

There's no such thing as an empty char. The closest you can get is `'\0'` , the Unicode "null" character. Given that you can embed that within string literals or express it on its own very easily, why would you want a separate field for it? Equally, the "it's easy to confuse `""` and `" "`" arguments don't apply for `'\0'` .

If you could give an example of where you'd want to use it and why you think it would be better, that might help...

edited Oct 27 '16 at 19:32
samis
**3,568** 6 22 51

answered Sep 8 '10 at 17:45
Jon Skeet
**1106k** 704 8050

2 Isn't the \0 the 'end of the byte array'-character? Or am I confusing with something else? – Bertvan Sep 8 '10 at 17:47

5 @Bertvan: Why would there be a *character* at the end of a *byte* array? It's used for "null terminated" strings though, yes. – Jon Skeet Sep 8 '10 at 17:48

29 Char.MinValue is better than '\0' – Aliostad Sep 8 '10 at 17:50

8 @Aliostad: Out of interest, if there was a similar field for `Int32.Zero` , would you use that instead of the literal 0? If not, what's the difference here? – Jon Skeet Sep 8 '10 at 18:17

7 @Adam, @Jon -- what is the code for bell? Or backspace better, think think... Or maybe instead of thinking it is just better to write Char.Backspace? Another reason -- you say it is better to write '0' for terminator, instead, say Char.Terminator, however it is not -- it is too easy to make a typo (fully compiled, see above), but try to write Char.Termnator. There are enough reasons for me to avoid non-checkable, raw values (space missions failed because of stupid typos like that). – greenoldman Sep 9 '10 at 6:19

74

A char, unlike a string, is a discrete thing with a fixed size. A string is really a container of chars.

So, Char.Empty doesn't really make sense in that context. If you have a char, it's not empty.

answered Sep 8 '10 at 17:46

Joe
**32.2k** 14 90 107

3 Exactly right. It makes sense to ask if a container is empty or not. It makes no sense to ask of a int or float or char is empty. – T.E.D. Sep 8 '10 at 18:10

1 @Joe: Then how can a string be empty if a string is a

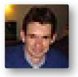
8 Because a string isn't the individual objects, it's the collection. Think of a bucket of rocks. I can't have an empty rock. But I can have an empty bucket. – Joe Jan 29 '13 at 17:42

2 I would phrase it as "a char is a primitive, value type, and a string is non-primitive, reference type". – samis Aug 26 '13 at 14:26

2 This is the real answer. – Gandalf458 Aug 10 '17 at 21:48

29

There's no such thing as an empty character. It always contains **something**. Even '\0' is a character.

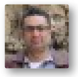
answered Sep 8 '10 at 17:46

Philippe Leybaert
**134k** 26 187 211

6 +1 nice channeling of sir jon! – kenny Sep 8 '10 at 19:20

23

Use `Char.MinValue` which works the same as '\0'. But be careful it is not the same as `String.Empty` .

edited Sep 28 '11 at 11:08

answered Sep 8 '10 at 17:48

Aliostad
**70.4k** 15 136 191

Thanks, haven't seen that before. Do you know if it work in myString.Replace('c', Char.MinValue)? I should give it a try. – Joan Venge Sep 8 '10 at 17:50

19

You could use nullable chars.

```
char? c
```

answered Sep 8 '10 at 17:54

paquetp
**1,458** 10 18

This allows ''? Or just null? – Joan Venge Sep 8 '10 at 17:57

In your case, you could do this: myString.Replace("c", (c == null ? "" : c.ToString())) – paquetp Sep 8 '10 at 18:13

10

If you don't need the entire string, you can take advantage of the delayed execution:

```
public static class StringExtensions
{
    public static IEnumerable<char> RemoveChar(this IEnume
char removingChar)
    {
        return originalString.Where(@char => @char != remo
    }
}
```

You can even combine multiple characters...

```
string veryLongText = "abcdefghijk...";

IEnumerable<char> firstFiveCharsWithoutCsAndDs = veryLongT
            .RemoveChar('c')
            .RemoveChar('d')
            .Take(5);
```

```
public static class StringExtensions
{
    public static IEnumerable<char> RemoveChars(this IEnum
        params char[] removingChars)
    {
        return originalString.Except(removingChars);
    }
}
```

and its usage:

```
        var veryLongText = "abcdefghijk...";
        IEnumerable<char> firstFiveCharsWithoutCsAndDs = v
            .RemoveChars('c', 'd')
            .Take(5)
            .ToArray(); //to prevent multiple execution of
```

edited Sep 9 '10 at 19:08

answered Sep 8 '10 at 19:17

Notoriousxl
**995** 12 25

1 Genius example. – Joan Venge Sep 8 '10 at 19:41

@Joan: thanks... even if "Genius" it's a bit exaggerated :P (I don't know about its performances when removingChars will become a big array...) – Notoriousxl Sep 8 '10 at 19:59

1 Yesterday I forgot: pay attention on how you are using the result variable "firstFiveCharsWithoutCsAndDs". If you don't want to pass it to another "yield" method (like those of LINQ), call immediately a ".ToArray()" after the "Take(5)"... otherwise, the "RemoveChars + Take" chain will be executed every time you access the variable in a "traditional" fashion (for example, every you call a "Count()" on it, or when you traverse it in a foreach without "yield return") – Notoriousxl Sep 9 '10 at 19:07

1 @nawfal efficiency-wise you're right, but I think that myString.Except("c") is more declarative than myString.Replace('c', '') :P (and it scales pretty well: myString.Except("aeiou")) – Notoriousxl Feb 5 '13 at 18:28

5

the same reason there isn't an `int.Empty` . Containers can be empty. Scalar values cannot be. If you mean 0 (which is not empty), then use `'\0'` . If you mean `null` , then use `null` :)

answered Sep 8 '10 at 17:51

tenfour
**27.4k** 12 61 123

2 null is not possible as char is a ValueType. You'd have to use char? to be able to assign null to it. – Femaref Sep 8 '10 at 17:53

you chould make it nullable. see my answer – paquetp Sep 8 '10 at 17:54

Good point man. – Joan Venge Sep 8 '10 at 17:56

5

A char is a value type, so its value cannot be null. (Unless it is wrapped in a Nullable container).

Since it can't be null, in contains some numeric code and each code is mapped to some character.

answered Sep 8 '10 at 17:53

epotter
**4,793** 6 56 84

```
myString = myString.Replace('c'.ToString(), "");
```

4

OK, this is not particularly elegant for removing letters, since the .Replace method has an overload that takes string parameters. But this works for removing carriage returns, line feeds, tabs, etc. This example removes tab characters:

```
myString = myString.Replace('\t'.ToString(), "");
```

edited May 21 '11 at 0:26

answered May 21 '11 at 0:12

Mike Taverne
**6,471** 2 22 41

---

3

Not an answer to your question, but to denote a default `char` you can use just

```
default(char)
```

which is same as `char.MinValue` which in turn is same as `\0`. One shouldn't use if for something like an empty string though.

answered Feb 4 '13 at 9:53

nawfal
**44.6k** 36 260 306

1

Doesn't answer your first question - but for the specific problem you had, you can just use strings instead of chars, right?:

```
 myString.Replace("c", "")
```

There a reason you wouldn't want to do that?

edited Jan 29 '13 at 16:32

answered Jan 29 '13 at 16:24

Ian Grainger
**2,927** 2 30 56

---

1

You can also rebuild your string character by character, excluding the characters that you want to get rid of.

Here's an extension method to do this:

```
    static public string RemoveAny(this string s, string cl
    {
        var result = "";
        foreach (var c in s)
            if (charsToRemove.Contains(c))
                continue;
            else
                result += c;

        return result;
    }
```

It's not slick or fancy, but it works well.

```
string newString = "My_String".RemoveAny("_"); //yields "M
```

answered Sep 2 '15 at 14:12

C. Tewalt
**1,463** 2 18 39

Use a `StringBuilder` for `result`. Why not wrap `return s.Replace(charsToRemove,"");` ? – aloisdg Jul 9 '16 at 15:26

0

How about BOM, the magical character Microsoft adds to start of files (at least XML)?

answered Sep 10 '10 at 5:02

Arto Viitanen
**174** 3

The wording on Wikipedia here is quite unfortunate; the BOM is not a character in this context. And what is your question exactly? :) – bzlm Oct 5 '10 at 11:36

@bzlm "how about..." ... – onemach Feb 21 '12 at 8:20

@onemach, so, whether `myString.Replace ('c', '')` could be achieved by `myString.Replace ('c', UTF_BOM)`. Then I'd say the answer is "how *not* about...". – bzlm Feb 21 '12 at 9:38

0

if you want to elliminate the empty char in string the following will work, just convert to any datatype representation you want. thanks,

```
        Int32 i;

        String name;

        Int32[] array_number = new int[100];

        name = "1 3  5  17   8    9    6";

        name = name.Replace(' ', 'x');

        char[] chr = name.ToCharArray();


        for (i = 0; i < name.Length; i++)
        {
            if ((chr[i] != 'x'))
            {
                array_number[i] = Convert.ToInt32(chr[i].T
                MessageBox.Show(array_number[i].ToString()
            }

        }

    }
```

edited Jun 27 '13 at 9:28

Joan Venge
**106k** 176 404 627

answered Jun 26 '13 at 13:48

Alexander Zaldostanov
**2,006** 3 23 31

---

0

In terms of C# language, the following may not make much sense. And this is not a direct answer to the question. But fowlloing is what I did in one of my business scenario

```
            Console.Write("Success");
        }
```

The `null` and `white space` had different business flows in my project. While inserting into database, I need to insert `empty string` to the database if it is white space.

answered Sep 5 '13 at 10:16

Lijo
**11.1k** 55 200 340

---

0

I know this one is pretty old, but I encountered an issue recently with having to do multiple replacements to make a file name safe. First, in the latest .NET string.Replace function null is the equivalent to empty character. Having said that, what is missing from .Net is a simple replace all that will replace any character in an array with the desired character. Please feel free to reference the code below (runs in LinqPad for testing).

```
// LinqPad .ReplaceAll and SafeFileName
void Main()
{

    ("a:B:C").Replace(":", "_").Dump();
character for one character => a_B_C
    ("a:B:C").Replace(":", null).Dump();
=> aBC
    ("a:B*C").Replace(":", null).Replace("*",null).Dump();
multiples

    // Need a ReplaceAll, so I don't have to chain calls


    ("abc/123.txt").SafeFileName().Dump();
    ("abc/1/2/3.txt").SafeFileName().Dump();
```

```
}


static class StringExtensions
{

    public static string SafeFileName(this string value, cl
    {
        return value.ReplaceAll(replacement, ':','*','?','
    }

    public static string ReplaceAll(this string value, char
charsToGo){

        if(replacement.HasValue == false){
                return string.Join("", value.AsEnumerable(
charsToGo.Contains(x) == false));
        }
        else{

            if(charsToGo.Contains(replacement.Value)){
                throw new ArgumentException(string.Format(
", replacement), "replacement");
            }

            return string.Join("", value.AsEnumerable().Se
charsToGo.Contains(x) == true ? replacement : x));
        }

    }

}
```

answered Feb 27 '15 at 15:18

Jim
**756** 1 7 12

If you want to remove characters that satisfy a specific condition, you may use this:

```
string s = "SoMEthInG";
s = new string(s.ToCharArray().Where(c => char.IsUpper(c))
```

(This will leave only the uppercase characters in the string.)

In other words, you may convert the string to an `IEnumerable<char>` , make changes on it and then convert it back to a string as shown above.

Again, this enables to not only remove a specific char because of the lambda expression, although you can do so if you change the lambda expression like this: `c => c != 't'` .

answered Jun 26 '16 at 16:49

florien
**195** 3 13

0

Easiest way to blanket remove a character from string is to Trim it

```
cl = cl.Trim(' ');
```

Removes all of the spaces in a string

answered Sep 4 '17 at 11:07

MrSmudge
**44** 1

This is helpful if one wants to use use .Replace('c', ' ') with the downside of removing other whitespaces. But its more helpful than lots of other answers given. – Jan Feb 27 '18 at 9:30

Yup, I was wrong on this one, good job – MrSmudge Jan 11 at 16:43

-1

use

```
myString.Replace ("c", "")
```

answered Jan 26 '16 at 6:30

Chintan
**100** 1 3

1 This is a duplicate of an answer from 2013 by Ian Grainger. – Joe Gayetty Apr 17 '18 at 14:34