

String vs. StringBuilder

Asked 16 years, 3 months ago Modified 11 months ago

Viewed 227k times



238

I understand the difference between `String` and `StringBuilder` (`StringBuilder` being mutable) but is there a large performance difference between the two?



The program I'm working on has a lot of case driven string appends (500+). Is using `StringBuilder` a better choice?



c#

.net

performance

Share

Improve this question

Follow

edited Jul 13, 2013 at 10:06



Peter Mortensen

31.6k ● 22 ● 109 ● 133

asked Sep 16, 2008 at 15:55



Kuvo

2,411 ● 2 ● 15 ● 10

24 Answers

Sorted by:

Highest score (default)



Yes, the performance difference is significant. See the KB article "[How to improve string concatenation performance in Visual C#](#)".

256



I have always tried to code for clarity first, and then optimize for performance later. That's much easier than doing it the other way around! However, having seen the enormous performance difference in my applications between the two, I now think about it a little more carefully.

Luckily, it's relatively straightforward to run performance analysis on your code to see where you're spending the time, and then to modify it to use `StringBuilder` where needed.

Share Improve this answer

Follow

edited Jul 12, 2014 at 19:16



[BartoszKP](#)

35.9k ● 15 ● 107 ● 133

answered Sep 16, 2008 at 15:59



[Jay Bazuzi](#)

46.4k ● 16 ● 115 ● 170


54 A good rule of thumb is to use Strings when you aren't going to change it and use StringBuilder if you will be changing it.

– [Tim](#) Sep 16, 2008 at 17:23

32 I like this answer a lot, especially the advice to code for clarity before performance. As developers, we spend as much or more time reading code as we do writing it.

– [Scott Lawrence](#) Sep 16, 2008 at 18:18

2 on basis of my experience if you trying to concat around 10-15 string then you can go with string but if no. of strings are more than that then use string builder – [Peeyush](#) Mar 1, 2012 at 6:38

5 It still depends on how one uses it, for a reference to actually tests I like [Coding Horror - The Sad Tragedy of Micro-Optimization Theater](#) – Erik Philips Jul 27, 2013 at 17:38 

1 If String builder is so powerful then why we have String in existence. Why don't we completely wipe out String. I asked this question so that you can tell me the BENEFIT of String over StringBuilder – [Unbreakable](#) May 24, 2018 at 18:56



To clarify what Gillian said about 4 string, if you have something like this:

62



```
string a, b, c, d;  
a = b + c + d;
```



then it would be faster using strings and the plus operator. This is because (like Java, as Eric points out), it internally uses StringBuilder automatically (Actually, it uses a primitive that StringBuilder also uses)

However, if what you are doing is closer to:

```
string a, b, c, d;  
a = a + b;  
a = a + c;  
a = a + d;
```

Then you need to explicitly use a StringBuilder. .Net doesn't automatically create a StringBuilder here, because it would be pointless. At the end of each line, "a" has to be an (immutable) string, so it would have to create and dispose a StringBuilder on each line. For

speed, you'd need to use the same `StringBuilder` until you're done building:

```
string a,b,c,d;  
StringBuilder e = new StringBuilder();  
e.Append(b);  
e.Append(c);  
e.Append(d);  
a = e.ToString();
```

Share Improve this answer

answered Sep 16, 2008 at 16:18

Follow



James Curran

103k ● 37 ● 185 ● 262

-
- 6 There is no reason why the C# compiler needs to treat the second sample differently from the first. In particular there is no obligation to produce a string at the end of each line. The compiler may behave as you say, but it's under no obligation to do so. – [CodesInChaos](#) Jul 27, 2013 at 20:23 ✎

@CodesInChaos, an immutable string variable is being specifically assigned to at the end of each line, doesn't that create the obligation to produce a string? However, I agree with the point that there's no reason to treat each individual line differently (and I'm not sure if it does), the performance loss comes from the reallocation though, so it wouldn't matter. – [Saeb Amini](#) Jan 22, 2015 at 12:11

-
- 1 @SaebAmini - If `a` is a local variable, and the object it references hasn't been assigned to some other variable (that might be accessible to another thread), a good optimizer could determine that `a` is not accessed *by any other code* during this sequence of lines; only the *final* value of `a` matters. So it could treat those three lines of code as if they were written `a = b + c + d;`. – [ToolmakerSteve](#) Nov 15, 2018 at 13:01 ✎
-



33



StringBuilder is preferable *IF* you are doing multiple loops, or forks in your code pass... however, for PURE performance, if you can get away with a *SINGLE* string declaration, then that is much more performant.

For example:

```
string myString = "Some stuff" + var1 + " more stuff"
                  + var2 + " other stuff" .... etc...
```

is more performant than



```
StringBuilder sb = new StringBuilder();
sb.Append("Some Stuff");
sb.Append(var1);
sb.Append(" more stuff");
sb.Append(var2);
sb.Append("other stuff");
// etc.. etc.. etc..
```

In this case, StringBuild could be considered more maintainable, but is not more performant than the single string declaration.

9 times out of 10 though... use the string builder.

On a side note: string + var is also more performant than the string.Format approach (generally) that uses a StringBuilder internally (when in doubt... check reflector!)



-
- 21 I wish you'd said how you know that / how to verify that.
– [ChrisW](#) Jul 17, 2009 at 1:39
-
- 2 You don't verify performance in reflector: You verify performance by timing release code, analyze with a profiler and seek explanation with reflector. – [Albin Sunnanbo](#) Oct 26, 2010 at 6:24
-
- 3 Consider using `String.Format()` to concatenate a small number of small strings, especially if the goal is formatting messages to show the user. – [Gary Kindel](#) Dec 18, 2010 at 15:39
-
- 1 This is very bad information. ("string myString is more performant") not true at all. – [Tom Stickel](#) Apr 4, 2013 at 4:03

-
- 3 The information in this answer is incorrect. `StringBuilder` is a little bit faster than concatenation done in the same statement; however, you will only notice a difference between the two if you do it hundreds of thousands of times ([Source](#)). As said in the source, "it just doesn't matter!"
– [David Sherret](#) Oct 7, 2014 at 17:57 
-



A simple example to demonstrate the difference in speed when using `String` concatenation vs `StringBuilder`:

33



```
System.Diagnostics.Stopwatch time = new Stopwatch();
string test = string.Empty;
time.Start();
for (int i = 0; i < 100000; i++)
{
    test += i;
}
```



```
time.Stop();
System.Console.WriteLine("Using String concatenation:
time.ElapsedMilliseconds + " milliseconds");
```

Result:

Using String concatenation: 15423 milliseconds

```
StringBuilder test1 = new StringBuilder();
time.Reset();
time.Start();
for (int i = 0; i < 100000; i++)
{
    test1.Append(i);
}
time.Stop();
System.Console.WriteLine("Using StringBuilder: " + tim
milliseconds");
```

Result:

Using StringBuilder: 10 milliseconds

As a result, the first iteration took 15423 ms while the second iteration using `StringBuilder` took 10 ms.

It looks to me that using `StringBuilder` is faster, a lot faster.

Share Improve this answer

edited Nov 8, 2018 at 15:21

Follow



lion

98 ● 5



-
- 2 It depends on how many times you're going to change the string. StringBuilder has overhead, so string is faster for limited concatenations. If you are going to append or modify thousands of times (usually not the case) then StringBuilder is faster in those scenarios. – [SendETHToThisAddress](#) Apr 9, 2021 at 20:04
-



This benchmark shows that regular concatenation is faster when combining 3 or fewer strings.

27



<http://www.chinhdo.com/20070224/stringbuilder-is-not-always-faster/>



StringBuilder can make a very significant improvement in memory usage, especially in your case of adding 500 strings together.



Consider the following example:

```
string buffer = "The numbers are: ";
for( int i = 0; i < 5; i++)
{
    buffer += i.ToString();
}
return buffer;
```

What happens in memory? The following strings are created:


```
1 - "The numbers are: "  
2 - "0"  
3 - "The numbers are: 0"  
4 - "1"  
5 - "The numbers are: 01"  
6 - "2"  
7 - "The numbers are: 012"  
8 - "3"  
9 - "The numbers are: 0123"  
10 - "4"  
11 - "The numbers are: 01234"  
12 - "5"  
13 - "The numbers are: 012345"
```

By adding those five numbers to the end of the string we created 13 string objects! And 12 of them were useless! Wow!

StringBuilder fixes this problem. It is not a "mutable string" as we often hear (*all strings in .NET are immutable*). It works by keeping an internal buffer, an array of char. Calling Append() or AppendLine() adds the string to the empty space at the end of the char array; if the array is too small, it creates a new, larger array, and copies the buffer there. So in the example above, StringBuilder might only need a single array to contain all 5 additions to the string-- depending on the size of its buffer. You can tell StringBuilder how big its buffer should be in the constructor.

Share Improve this answer


answered Sep 16, 2008 at 16:27

Follow



Matt Trunnell

271 ● 2 ● 3

-
- 2 Minor nit: its a bit odd to say "we created 13 string objects, and 12 of them were useless", and then to say StringBuilder fixes this problem. After all, *six* of the strings you have no choice but to create; they are from `i.ToString()` . So with StringBuilder, you still have to create 6 + 1 strings; it reduced creating 13 strings to creating 7 strings. But that is still the wrong way to look at it; the creation of the six number strings is not relevant. Bottom line: you simply shouldn't have mentioned the six strings created by `i.ToString()` ; they aren't part of the efficiency comparison. – [ToolmakerSteve](#)
Nov 15, 2018 at 13:17 
-



String Vs String Builder:

23

First thing you have to know that In which assembly these two classes lives?



So,



string is present in `System` namespace.



and

StringBuilder is present in `System.Text` namespace.

For **string** declaration:

You have to include the `System` namespace. something like this. `Using System;`

and

For **StringBuilder** declaration:

You have to include the `System.text` namespace.
something like this. `Using System.text;`

Now Come the the actual Question.

What is the differene between **string** & **StringBuilder**?

The main difference between these two is that:

string is immutable.

and

StringBuilder is mutable.

So Now lets discuss the difference between *immutable* and *mutable*

Mutable: : means Changable.

Immutable: : means Not Changable.

For example:

```
using System;

namespace StringVsStrigBuilder
{
    class Program
    {
        static void Main(string[] args)
        {
            // String Example

            string name = "Rehan";
            name = name + "Shah";
            name = name + "RS";
        }
    }
}
```

```

        name = name + "---";
        name = name + "I love to write programs.";

        // Now when I run this program this output
        // output : "Rehan Shah RS --- I love to w
    }
}
}

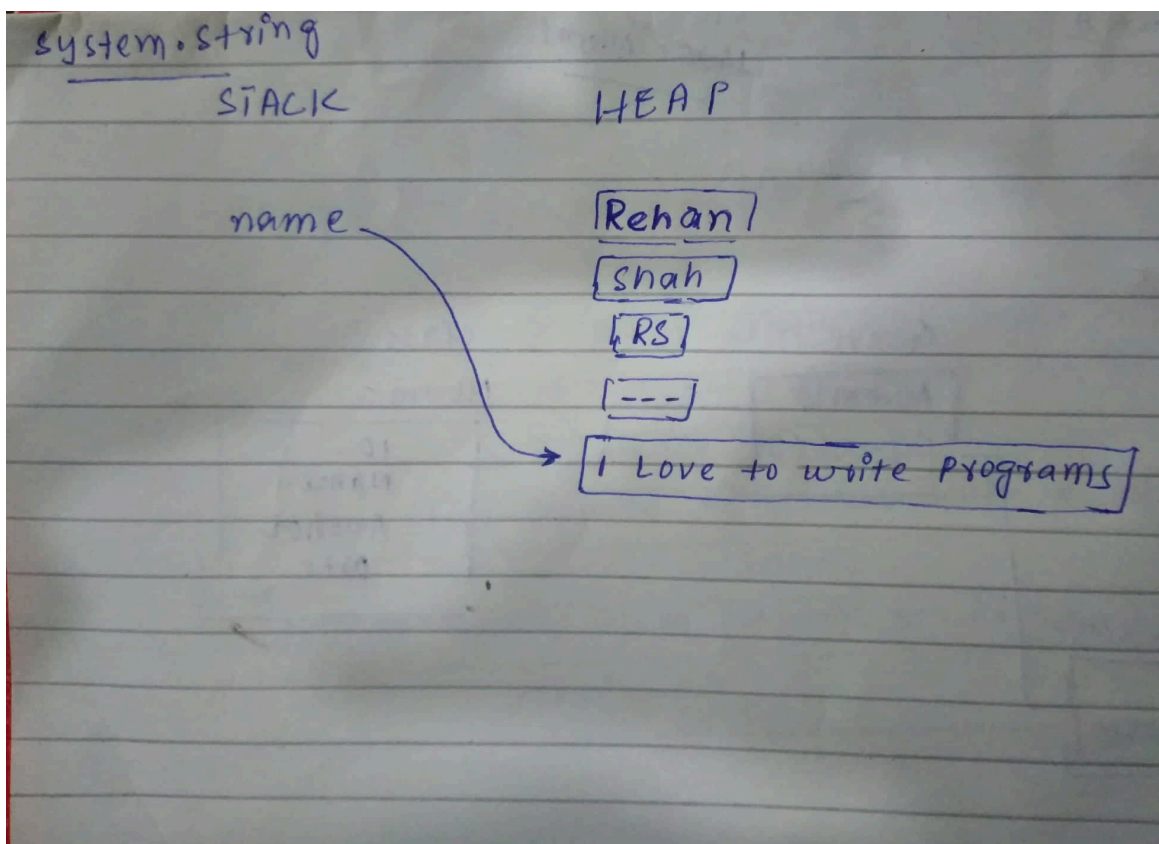
```

So in this case we are going to changing same object 5-times.

So the Obvious question is that ! What is actually happen under the hood, when we change the same string 5-times.

This is What Happen when we change the same string 5-times.

let look at the figure.



Explanation:

When we first initialize this variable "name" to "Rehan" i-e `string name = "Rehan"` this variable get created on stack "name" and pointing to that "Rehan" value. after this line is executed: `name = name + "Shah"`. the reference variable is no longer pointing to that object "Rehan" it now pointing to "Shah" and so on.

So `string` is immutable meaning that once we create the object in the memory we can't change them.

So when we concatenating the `name` variable the previous object remains there in the memory and another new string object is get created...

So from the above figure we have five-objects the four-objects are thrown away they are not used at all. They still remain in memory and they occupy the amount of memory. "Garbage Collector" is responsible for that so clean that resources from the memory.

So in case of string anytime when we manipulate the string over and over again we have some many objects Created and stay there at in the memory.

So this is the story of string Variable.

Now Let's look at toward StringBuilder Object. **For**

Example:

```
using System;  
using System.Text;
```

```

namespace StringVsStrigBuilder
{
    class Program
    {
        static void Main(string[] args)
        {
            // StringBuilder Example

            StringBuilder name = new StringBuilder();
            name.Append("Rehan");
            name.Append("Shah");
            name.Append("RS");
            name.Append("---");
            name.Append("I love to write programs.");

            // Now when I run this program this output
            // output : "Rehan Shah Rs --- I love to w

        }
    }
}

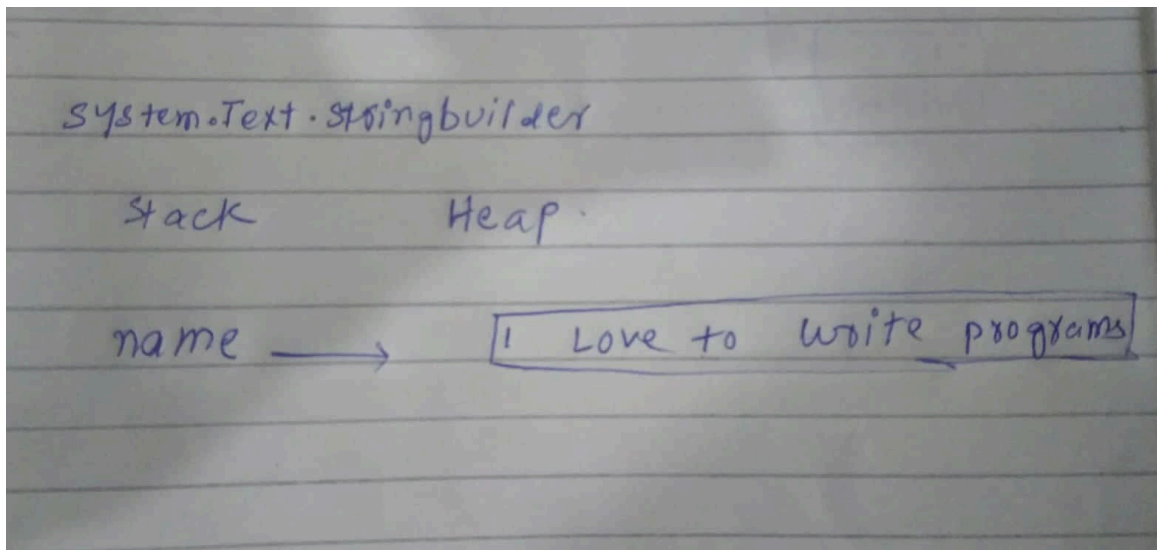
```

So in this case we are going to changing same object 5-times.

So the Obvious question is that ! What is actually happen under the hood, when we change the same StringBuilder 5-times.

This is What Happen when we change the same StringBuilder 5-times.

let look at the figure.



Explanation: In case of StringBuilder object. you wouldn't get the new object. The same object will be change in memory so even if you change the object et say 10,000 times we will still have only one stringBuilder object.

You don't have alot of garbage objects or non_referenced stringBuilder objects because why it can be change. It is mutable meaning it change over a time?

Differences:

- String is present in System namespace where as Stringbuilder present in System.Text namespace.
- string is immutable where as StringBuilder is mutabe.

Share Improve this answer

answered Dec 15, 2018 at 16:37

Follow



Rehan Shah

1,627 ● 16 ● 32



13



Yes, `StringBuilder` gives better performance while performing repeated operation over a string. It is because all the changes are made to a single instance so it can save a lot of time instead of creating a new instance like `String`.

String Vs StringBuilder

- `String`
 1. under `System` namespace
 2. immutable (read-only) instance
 3. performance degrades when continuous change of value occurs
 4. thread safe
- `StringBuilder` (mutable string)
 1. under `System.Text` namespace
 2. mutable instance
 3. shows better performance since new changes are made to existing instance

Strongly recommend dotnet mob article : [String Vs StringBuilder in C#](#).

Related Stack Overflow question: [Mutability of string when string doesn't change in C#?](#).

answered Feb 26, 2015 at 17:52



Shamseer K

5,271 ● 2 ● 29 ● 46



8



StringBuilder reduces the number of allocations and assignments, at a cost of extra memory used. Used properly, it can completely remove the need for the compiler to allocate larger and larger strings over and over until the result is found.



```
string result = "";
for(int i = 0; i != N; ++i)
{
    result = result + i.ToString();    // allocates a new string
    // to result, which gets repeated N times
}
```

VS.

```
String result;
StringBuilder sb = new StringBuilder(10000);    // create a buffer
for(int i = 0; i != N; ++i)
{
    sb.Append(i.ToString());                // fill the buffer
    // until it overflows the buffer
}


result = sb.ToString();    // assigns once
```

Follow



Harsh
45 ● 8

answered Sep 16, 2008 at 16:10



moswald
11.7k ● 7 ● 56 ● 78



6



The performance of a concatenation operation for a String or StringBuilder object depends on how often a memory allocation occurs. A String concatenation operation always allocates memory, whereas a StringBuilder concatenation operation only allocates memory if the StringBuilder object buffer is too small to accommodate the new data. Consequently, the String class is preferable for a concatenation operation if a fixed number of String objects are concatenated. In that case, the individual concatenation operations might even be combined into a single operation by the compiler. A StringBuilder object is preferable for a concatenation operation if an arbitrary number of strings are concatenated; for example, if a loop concatenates a random number of strings of user input.

Source: [MSDN](#)

Share Improve this answer

edited Dec 11, 2011 at 21:06

Follow



sjngm

12.9k ● 16 ● 88 ● 117

answered Oct 26, 2010 at 0:36



user487069

71 ● 1 ● 1



`StringBuilder` is better for building up a string from many non-constant values.

4



If you're building up a string from a lot of constant values, such as multiple lines of values in an HTML or XML document or other chunks of text, you can get away with just appending to the same string, because almost all compilers do "constant folding", a process of reducing the parse tree when you have a bunch of constant manipulation (it's also used when you write something like `int minutesPerYear = 24 * 365 * 60`). And for simple cases with non-constant values appended to each other, the .NET compiler will reduce your code to something similar to what `StringBuilder` does.

But when your append can't be reduced to something simpler by the compiler, you'll want a `StringBuilder`. As fizch points out, that's more likely to happen inside of a loop.

Share Improve this answer

Follow

edited Mar 16, 2016 at 17:12



David Ferenczy
Rogożan

25.4k ● 11 ● 85 ● 73

answered Sep 16, 2008 at 20:30



JasonTrue

19.6k ● 5 ● 37 ● 61



4

Consider '[The Sad Tragedy of Micro-Optimization Theater](#)'.

Share Improve this answer

edited Mar 4, 2023 at 13:18



Follow



Andrew D. Bond

1,250 ● 1 ● 16 ● 12



answered Jun 18, 2010 at 15:03



Jim G.

15.4k ● 22 ● 108 ● 176



2

Further to the previous answers, the first thing I always do when thinking of issues like this is to create a small test application. Inside this app, perform some timing test for both scenarios and see for yourself which is quicker.



IMHO, appending 500+ string entries should definitely use StringBuilder.



Share Improve this answer

answered Sep 16, 2008 at 16:02



RichS

3,147 ● 31 ● 27



1

I believe StringBuilder is faster if you have more than 4 strings you need to append together. Plus it can do some cool things like AppendLine.



Share Improve this answer

answered Sep 16, 2008 at 15:59

Follow



Gilligan

1,515 ● 2 ● 15 ● 16



1

In .NET, StringBuilder is still faster than appending strings. I'm pretty sure that in Java, they just create a StringBuffer under the hood when you append strings, so there's isn't really a difference. I'm not sure why they haven't done this in .NET yet.



Share Improve this answer

answered Sep 16, 2008 at 15:59

Follow



Eric Z Beard

38.4k ● 27 ● 101 ● 147



1

`StringBuilder` is significantly more efficient but you will not see that performance unless you are doing a large amount of string modification.



Below is a quick chunk of code to give an example of the performance. As you can see you really only start to see a major performance increase when you get into large iterations.



As you can see the 200,000 iterations took 22 seconds while the 1 million iterations using the `StringBuilder` was almost instant.

```
string s = string.Empty;  
StringBuilder sb = new StringBuilder();
```

```

Console.WriteLine("Beginning String + at " + DateTime.

for (int i = 0; i <= 50000; i++)
{
    s = s + 'A';
}

Console.WriteLine("Finished String + at " + DateTime.N
Console.WriteLine();

Console.WriteLine("Beginning String + at " + DateTime.

for (int i = 0; i <= 200000; i++)
{
    s = s + 'A';
}

Console.WriteLine("Finished String + at " + DateTime.N
Console.WriteLine();
Console.WriteLine("Beginning Sb append at " + DateTime

for (int i = 0; i <= 1000000; i++)
{
    sb.Append("A");
}
Console.WriteLine("Finished Sb append at " + DateTime.

Console.ReadLine();

```

Result of the above code:

Beginning String + at 28/01/2013 16:55:40.

Finished String + at 28/01/2013 16:55:40.

Beginning String + at 28/01/2013 16:55:40.

Finished String + at 28/01/2013 16:56:02.

Beginning Sb append at 28/01/2013 16:56:02.

Finished Sb append at 28/01/2013 16:56:02.

Share Improve this answer

edited Jun 20, 2020 at 9:12

Follow



Community Bot

1 • 1

answered Jan 28, 2013 at 17:01



CathalMF

10.1k • 7 • 74 • 113

-
- 1 If String builder is so powerful then why we have String in existence. Why don't we completely wipe out String. I asked this question so that you can tell me the BENEFIT of String over StringBuilder – [Unbreakable](#) May 24, 2018 at 18:55
-

Thread safety, memory reduction, and functional (as in functional programming) – [feyd](#) Nov 10, 2020 at 15:08



Using strings for concatenation can lead to a runtime complexity on the order of $O(n^2)$.

1



If you use a `StringBuilder`, there is a lot less copying of memory that has to be done. With the `StringBuilder(int capacity)` you can increase performance if you can estimate how large the final `String` is going to be. Even if you're not precise, you'll probably only have to grow the capacity of `StringBuilder` a couple of times which can help performance also.



Share Improve this answer

Follow

edited Mar 16, 2016 at 17:13



David Ferenczy
Rogożan

25.4k ● 11 ● 85 ● 73

answered Sep 16, 2008 at 16:08



Steve g

2,489 ● 17 ● 17



1



I have seen significant performance gains from using the `EnsureCapacity(int capacity)` method call on an instance of `StringBuilder` before using it for any string storage. I usually call that on the line of code after instantiation. It has the same effect as if you instantiate the `StringBuilder` like this:

```
var sb = new StringBuilder(int capacity);
```

This call allocates needed memory ahead of time, which causes fewer memory allocations during multiple `Append()` operations. You have to make an educated guess on how much memory you will need, but for most applications this should not be too difficult. I usually err on the side of a little too much memory (we are talking 1k or so).

Share Improve this answer

Follow

edited Mar 16, 2016 at 17:20



David Ferenczy
Rogożan

25.4k ● 11 ● 85 ● 73

answered Sep 16, 2008 at 17:05



[Jason Jackson](#)

17.2k ● 8 ● 51 ● 75

There is no need to call `EnsureCapacity` just after the `StringBuilder` instantiation. Simply instantiate the `StringBuilder` like this: `var sb = new StringBuilder(int capacity) .`

– [David Ferenczy Rogożan](#) Mar 16, 2016 at 17:16

I was told that it helps to use a prime number. – [Karl Gjertsen](#) May 9, 2016 at 8:46

I think prime numbers relates to initial size of Dictionaries, not `StringBuilders`, because Dictionaries have a hash table behind the scenes and now this begins to matter due to hash function details. However, I think .NET has abstracted away that need (compared to MFC) and automatically "primeifies" your given capacity behind the scenes. – [Jonas](#) Sep 5, 2023 at 14:18



0

If you're doing a lot of string concatenation, use a `StringBuilder`. When you concatenate with a `String`, you create a new `String` each time, using up more memory.



Alex

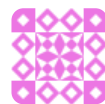


Share Improve this answer

answered Sep 16, 2008 at 16:01



Follow



[Alex Fort](#)

18.8k ● 5 ● 44 ● 51



`String` and `StringBuilder` are actually both immutable, the `StringBuilder` has built in buffers which allow its size to be

0



managed more efficiently. When the StringBuilder needs to resize is when it is re-allocated on the heap. By default it is sized to 16 characters, you can set this in the constructor.

eg.

```
StringBuilder sb = new StringBuilder(50);
```

Share Improve this answer

answered Sep 16, 2008 at 16:13

Follow



capgpilk

83 ● 1 ● 8

3 Not sure that you understand what immutable means. Strings are immutable, they CANNOT be changed. Anything "changed" is actually just that the old value remains on the heap without ANY pointer to locate it. StringBuilder (mutable) is reference type on heap, pointer to the heap are changed and allocation of space for this change. – [Tom Stickel](#) Jun 9, 2014 at 22:32



0



String concatenation will cost you more. In Java, You can use either StringBuffer or StringBuilder based on your need. If you want a synchronized, and thread safe implementation, go for StringBuffer. This will be faster than the String concatenation.

If you do not need synchronized or Thread safe implementation, go for StringBuilder. This will be faster than String concatenation and also faster than StringBuffer as there is no synchronization overhead.

Share Improve this answer

answered Sep 16, 2008 at 18:09

Follow



[raffimd](#)

49 ● 4



0

My approach has always been to use StringBuilder when concatenating 4 or more strings OR When I don't know how many concatenations are to take place.



[Good performance related article on it here](#)



Share Improve this answer

edited Dec 1, 2009 at 15:02



Follow



[Peter Mortensen](#)

31.6k ● 22 ● 109 ● 133

answered Sep 16, 2008 at 16:11



JohnC



-1

StringBuilder will perform better, from a memory standpoint. As for processing, the difference in time of execution may be negligible.



Share Improve this answer

answered Sep 16, 2008 at 15:57

Follow



[DevelopingChris](#)

40.7k ● 30 ● 89 ● 119



StringBuilder is probably preferable. The reason is that it allocates more space than currently needed (you set the number of characters) to leave room for future appends.

-1



Then those future appends that fit in the current buffer don't require any memory allocation or garbage collection, which can be expensive. In general, I use `StringBuilder` for complex string concatenation or multiple formatting, then convert to a normal `String` when the data is complete, and I want an immutable object again.

[Share](#) [Improve this answer](#)

answered Sep 16, 2008 at 16:01

[Follow](#)



[deemer](#)

1,144 ● 8 ● 10



-1



As a general rule of thumb, if I have to set the value of the string more than once, or if there are any appends to the string, then it needs to be a string builder. I have seen applications that I have written in the past before learning about string builders that have had a huge memory footprint that just seems to keep growing and growing. Changing these programs to use the string builder cut down the memory usage significantly. Now I swear by the string builder.

[Share](#) [Improve this answer](#)

edited Dec 1, 2009 at 15:00

[Follow](#)



[Peter Mortensen](#)

31.6k ● 22 ● 109 ● 133

answered Sep 16, 2008 at 18:13



[user13288](#)

9 ● 1



Highly active question. Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.