**StringBuilder**

Documentazione di Microsoft Visual Studio

**Utilizzo della classe StringBuilder**

Questa classe rappresenta un oggetto simile a una stringa il cui valore è una sequenza modificabile di caratteri. Il valore può essere modificato dopo la creazione aggiungendo, rimuovendo, sostituendo, o inserendo i caratteri. Per confronti, vedere la classe String.

La capacità possibile di un'istanza StringBuilder è il numero massimo di caratteri che l'istanza può memorizzare in un determinato momento. La capacità è maggiore di, o uguale a, la lunghezza della rappresentazione di stringa del valore dell'istanza. La capacità può essere aumentata o diminuita tramite la proprietà Capacity o il metodo EnsureCapacity ma non può essere inferiore al valore della proprietà Length.

Se non si specifica la capacità o la capacità massima quando si inizializza un'istanza StringBuilder, vengono utilizzati i valori predefiniti specifici (16 caratteri).

# Impostazione di capacità e lunghezza

Sebbene l'oggetto **StringBuilder** sia un oggetto dinamico, che consente cioè di espandere il numero di caratteri della stringa incapsulata, è possibile specificare il valore massimo dei caratteri che la stringa può contenere. Questo valore è detto capacità dell'oggetto e non deve essere confuso con la lunghezza della stringa contenuta nell'oggetto **StringBuilder** corrente. È possibile ad esempio creare una nuova istanza della classe **StringBuilder** con la stringa "Hello", che ha lunghezza pari a 5, e specificare che l'oggetto deve avere una capacità massima di 25. Se non specificata esplicitamente la capacità di default è 16. Quando si modifica l'oggetto StringBuilder, le sue dimensioni non vengono modificate se non viene raggiunta la capacità specificata. Quando questo accade, il nuovo spazio viene allocato automaticamente e la capacità raddoppiata. È possibile specificare la capacità della classe **StringBuilder** utilizzando uno dei costruttori di overload. Nell'esempio che segue è specificato che l'oggetto MyStringBuilder può essere ampliato fino a un massimo di 25 spazi.

StringBuilder MyStringBuilder = new StringBuilder("Hello World!", 25);

Oppure

StringBuilder MyStringBuilder = new StringBuilder(25);

Inoltre, è possibile utilizzare la proprietà in lettura/scrittura Capacityper impostare la lunghezza massima dell'oggetto. Nell'esempio riportato di seguito viene utilizzata la proprietà **Capacity** per definire la lunghezza massima dell'oggetto.

MyStringBuilder.Capacity = 25;

### Funzionalità di StringBuilder

La dimensione corrente di un oggetto StringBuilder è definita dalla proprietà Length. È possibile accedere ai caratteri del valore di un oggetto StringBuilder utilizzando la proprietà Chars. Le posizioni di indice iniziano da zero.

La classe StringBuilder ha metodi che possono ridurre la dimensione dell'istanza corrente. Il metodo Clear rimuove tutti i caratteri e imposta la proprietà Length a zero. Il metodo Remove rimuove un intervallo di caratteri.

La classe StringBuilder include anche metodi con i quali è possibile espandere l'istanza corrente. I metodi AppendLine e Append aggiungono testo mentre il metodo Insert ne permette l’inserimento in una posizione specificata. Il metodo AppendFormat utilizza la funzionalità formattazione composita per aggiungere il testo formattato alla fine di un oggetto StringBuilder.

Il metodo Replace sostituisce tutte le occorrenze di un carattere o di una stringa nell'intero oggetto StringBuilder o in un intervallo di caratteri specificati.

È necessario convertire l'oggetto StringBuilder in un oggetto String prima di poter passare la stringa rappresentata dall'oggetto StringBuilder a un metodo contenente un parametro String o di visualizzarla nell'interfaccia utente. Eseguire questa conversione chiamando il metodo ToString. Es. MyStringBuilder.ToString()

### Considerazioni sulle prestazioni

L'operazione di concatenazione dell'oggetto String crea sempre un nuovo oggetto determinato dalla stringa esistente e dai nuovi dati. L'oggetto StringBuilder dispone di un buffer per includere la concatenazione dei nuovi dati. I nuovi dati vengono aggiunti al buffer se lo spazio disponibile; in caso contrario, un nuovo, più grande buffer viene allocato, i dati dal buffer di origine vengono copiati nel nuovo buffer e i nuovi dati quindi vengono aggiunti al nuovo buffer.

Le prestazioni delle operazioni di concatenazione per un oggetto StringBuilder o String dipendono dalla frequenza delle allocazioni di memoria. Un'operazione di concatenazione String assegna sempre la memoria, mentre un'operazione di concatenazione StringBuilder alloca memoria solo se il buffer dell'oggetto StringBuilder è troppo piccolo per inserire i nuovi dati. Utilizzare la classe String se si concatenano un numero fisso di oggetti String. In tal caso, il compilatore può combinare singole operazioni di concatenazione in un'unica operazione. Utilizzare un oggetto StringBuilder se si concatenano un numero arbitrario di stringhe, ad esempio, se si utilizza un ciclo per concatenare un numero casuale di sequenze di input.

### Allocazione di memoria

La capacità predefinita per questa implementazione è 16 e la capacità minima predefinita è Int32.MaxValue.

Un oggetto StringBuilder alloca più memoria quando il valore dell'istanza aumenta in misura maggiore della capacità che viene quindi modificata di conseguenza.

# Modifica della stringa StringBuilder

Nella tabella riportata di seguito sono elencati i metodi utilizzabili per modificare il contenuto di un oggetto **StringBuilder**.

|  |  |
| --- | --- |
| **Nome del metodo** | **Utilizzo** |
| StringBuilder.Append(…) | Consente di aggiungere informazioni alla fine dell'oggetto **StringBuilder** corrente. |
| StringBuilder.AppendFormat | Consente di sostituire un identificatore di formato passato in una stringa con testo formattato. |
| StringBuilder.AppendLine([string]) | Consente di aggiungere il carattere di terminazione della riga predefinito alla fine dell'oggetto **StringBuilder** corrente. |
| StringBuilder.Insert | Consente di inserire una stringa o un oggetto in corrispondenza dell'indice specificato dell'oggetto **StringBuilder** corrente. |
| StringBuilder.Remove | Consente di rimuovere un numero specificato di caratteri dall'oggetto **StringBuilder** corrente. |
| StringBuilder.Replace | Consente di sostituire caratteri all'interno dell'oggetto StringBuilder con un altro carattere specificato. |

### Append

Il metodo **Append** può essere utilizzato per aggiungere testo o una rappresentazione in forma di stringa di un oggetto alla fine di una stringa rappresentata dall'oggetto **StringBuilder** corrente. Nell'esempio che segue un oggetto **StringBuilder** viene inizializzato con "Hello World" e quindi viene aggiunto testo alla fine dell'oggetto stesso. Lo spazio viene allocato automaticamente secondo le necessità.

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

MyStringBuilder.Append(" What a beautiful day.");

Console.WriteLine(MyStringBuilder);

Nella console verrà visualizzato Hello World! What a beautiful day..

### AppendFormat

Il metodo **AppendFormat** aggiunge testo alla fine dell'oggetto **StringBuilder**, ma consente anche di implementare l'interfaccia **IFormattable** e quindi di accettare le stringhe di formato standard descritte nella sezione Formattazione dei tipi. È possibile utilizzare questo metodo per personalizzare il formato delle variabili e aggiungere tali valori a un oggetto **StringBuilder**. Nell'esempio riportato di seguito viene utilizzato il metodo **AppendFormat** per collocare un valore integer formattato come valore di valuta alla fine di un oggetto **StringBuilder**.

int MyInt = 25;

StringBuilder MyStringBuilder = new StringBuilder("Your total is ");

MyStringBuilder.AppendFormat("{0:C} ", MyInt);

Console.WriteLine(MyStringBuilder);

Nella console verrà visualizzato Your total is $25.00.

### AppendLine([string])

Il metodo **AppendLine** accoda una copia della stringa specificata seguita dal carattere di terminazione della riga predefinito alla fine dell'oggetto StringBuilder corrente.

### AppendFormat

Il metodo **AppendFormat** aggiunge testo alla fine dell'oggetto **StringBuilder**, ma consente anche di implementare l'interfaccia **IFormattable** e quindi di accettare le stringhe di formato standard descritte nella sezione Formattazione dei tipi. È possibile utilizzare questo metodo per personalizzare il formato delle variabili e aggiungere tali valori a un oggetto **StringBuilder**. Nell'esempio riportato di seguito viene utilizzato il metodo **AppendFormat** per collocare un valore integer formattato come valore di valuta alla fine di un oggetto **StringBuilder**.

int MyInt = 25;

StringBuilder MyStringBuilder = new StringBuilder("Your total is ");

MyStringBuilder.AppendFormat("{0:C} ", MyInt);

Console.WriteLine(MyStringBuilder);

Nella console verrà visualizzato Your total is $25.00.

### Insert

Il metodo **Insert** consente di aggiungere una stringa o un oggetto in una posizione specificata nell'oggetto **StringBuilder** corrente. Nell'esempio che segue viene utilizzato questo metodo per inserire una parola nella sesta posizione di un oggetto **StringBuilder**.

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

MyStringBuilder.Insert(6,"Beautiful ");

Console.WriteLine(MyStringBuilder);

Nella console verrà visualizzato Hello Beautiful World!.

### Remove

È possibile utilizzare il metodo **Remove** per rimuovere un numero specificato di caratteri dall'oggetto **StringBuilder** corrente, a partire da un indice a base zero specificato. Nell'esempio che segue viene utilizzato il metodo **Remove** per abbreviare un oggetto **StringBuilder**.

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

MyStringBuilder.Remove(5,7);

Console.WriteLine(MyStringBuilder);

Nella console verrà visualizzato Hello.

### Replace

Il metodo **Replace** può essere utilizzato per sostituire caratteri all'interno dell'oggetto **StringBuilder** con un altro carattere specificato. Nell'esempio riportato di seguito si utilizza il metodo **Replace** per cercare in un oggetto **StringBuilder** tutte le istanze del carattere punto esclamativo (!) e sostituirle con il carattere punto di domanda (?).

StringBuilder MyStringBuilder = new StringBuilder("Hello World!");

MyStringBuilder.Replace('!', '?');

Console.WriteLine(MyStringBuilder);

Nella console verrà visualizzato Hello World?.

**Nota:** Con StringBuilder è possible modificare il singolo carattere:

StringBuilder MySB = new StringBuilder("Prava", 50);

MySB[2] = 'o'; // Con StringBuilder Funziona!!!

* **Utilizzo di StringBuilder**

La classe **StringBuilder** crea un buffer di stringhe che offre prestazioni migliori se il programma esegue numerose modifiche di stringhe. La stringa **StringBuilder** consente inoltre di riassegnare singoli caratteri, operazione non supportata dal tipo di dati di stringa incorporato. Tramite il codice riportato di seguito, ad esempio, è possibile modificare il contenuto di una stringa senza crearne una nuova:

System.Text.StringBuilder sb = new System.Text.StringBuilder("Rat: the ideal pet");

sb[0] = 'C';

System.Console.WriteLine(sb.ToString());

System.Console.ReadLine();

Nell'esempio riportato di seguito, viene utilizzato un oggetto **StringBuilder**:

using System;

using System.Text;

public sealed class App

{

static void Main()

{

// Create a StringBuilder that expects to hold 50 characters.

// Initialize the StringBuilder with "ABC".

StringBuilder sb = new StringBuilder("ABC", 50);

// Append three characters (D, E, and F) to the end of the StringBuilder.

sb.Append(new char[] { 'D', 'E', 'F' });

// Append a format string to the end of the StringBuilder.

sb.AppendFormat("GHI{0}{1}", 'J', 'k');

// Display the number of characters in the StringBuilder and its string.

Console.WriteLine("{0} chars: {1}", sb.Length, sb.ToString());

// Insert a string at the beginning of the StringBuilder.

sb.Insert(0, "Alphabet: ");

// Replace all lowercase k's with uppercase K's.

sb.Replace('k', 'K');

// Display the number of characters in the StringBuilder and its string.

Console.WriteLine("{0} chars: {1}", sb.Length, sb.ToString());

}

}

// This code produces the following output.

//

// 11 chars: ABCDEFGHIJk

// 21 chars: Alphabet: ABCDEFGHIJK