**string e System.String**

Documentazione di Microsoft Visual Studio

Una stringa è un oggetto di tipo String il cui valore è testo. Internamente, il testo viene archiviato come raccolta di sola lettura sequenziale di oggetti Char. Le stringhe C# non presentano un carattere di terminazione null alla fine, pertanto una stringa C# può contenere qualsiasi numero di caratteri null incorporati ('\0'). La proprietà Length di una stringa rappresenta il numero di oggetti Char in essa contenuti e non il numero di caratteri Unicode. Per accedere ai singoli punti di codice Unicode in una stringa, utilizzare l'oggetto StringInfo.

**string e System.String**

In C#, la parola chiave string è un alias per String. String e string sono pertanto equivalenti ed è possibile utilizzare la convenzione di denominazione che si preferisce. La classe String fornisce molti metodi per creare, modificare e confrontare stringhe in modo sicuro. Inoltre, il linguaggio C# esegue l'overload di alcuni operatori per semplificare le operazioni comuni sulle stringhe. Per ulteriori informazioni sulla parola chiave, vedere string (Riferimenti per C#). Per ulteriori informazioni sul tipo e i relativi metodi, vedere String.

È possibile dichiarare e inizializzare stringhe in vari modi, come mostrato nell'esempio seguente:

// Declare without initializing.

string message1;

// Initialize to null.

string message2 = null;

// Initialize as an empty string.

// Use the Empty constant instead of the literal "".

string message3 = System.String.Empty;

//Initialize with a regular string literal.

string oldPath = "c:\\Program Files\\Microsoft Visual Studio 8.0";

// Initialize with a verbatim string literal.

string newPath = @"c:\Program Files\Microsoft Visual Studio 9.0";

// Use System.String if you prefer.

System.String greeting = "Hello World!";

// In local variables (i.e. within a method body)

// you can use implicit typing.

var temp = "I'm still a strongly-typed System.String!";

// Use a const string to prevent 'message4' from

// being used to store another string value.

const string message4 = "You can't get rid of me!";

Inizializzare una stringa con il valore costante Empty per creare un nuovo oggetto String la cui stringa è di lunghezza zero. La rappresentazione del valore letterale stringa di una stringa di lunghezza zero è"".

Usare il costruttore String solo quando si crea una string da char[], char

Costruttori della classe String:

|  |  |
| --- | --- |
| Nome | Descrizione |
| String(Char[]) | Consente di inizializzare una nuova istanza della classe String sul valore indicato da una matrice di caratteri Unicode. |
| String(Char, Int32) | Consente di inizializzare una nuova istanza della classe String sul valore indicato da uno specifico carattere Unicode ripetuto un numero precisato di volte. |
| String(Char[], Int32, Int32) | Consente di inizializzare una nuova istanza della classe String sul valore indicato da una matrice di caratteri Unicode, da una posizione iniziale del carattere all'interno di tale matrice e da una lunghezza. |

Esempio:

String myString = new String('A', 20);

Console.WriteLine("Stringa {0} di {1} caratteri", myString, myString.Length);

Output: Stringa AAAAAAAAAAAAAAAAAAAA di 20 caratteri

Esempio:

char[] letters = { 'A', 'B', 'C' };

string alphabet = new string(letters);

**Operatore somma**

L'operatore + concatena le stringhe:

**string** a = "good " + "morning";

Viene creato un oggetto stringa che contiene "good morning".

**Accesso al singolo carattere**

È possibile utilizzare l'operatore [] per accedere ai singoli caratteri di un tipo **string**:

Esempio:

**string** str = "test";

char x = str[2]; // x = 's';

Esempio:

string s5 = "Printing backwards";

for (int i = 0; i < s5.Length; i++)

{

System.Console.Write(s5[s5.Length - i - 1]); // outputs "sdrawkcab gnitnirP"

}

**Immutabilità degli oggetti stringa**

Gli oggetti stringa sono non modificabili, ovvero non possono essere modificati una volta creati. Tutti i metodi String e gli operatori C# che sembrano modificare una stringa, in realtà restituiscono i risultati in un nuovo oggetto stringa. Nell'esempio seguente, quando il contenuto di s1 e s2 viene concatenato per formare un'unica stringa, le due stringhe originali restano immutate. L'operatore += crea una nuova stringa che contiene il contenuto delle due stringhe combinato. Il nuovo oggetto viene assegnato alla variabile s1e l'oggetto originale assegnato a s1 viene rilasciato per l'operazione di Garbage Collection perché nessun'altra variabile contiene un riferimento a tale oggetto..

**Valori letterali stringa normali e verbatim (Caratteri di escape)**

Utilizzare valori letterali stringa normali quando è necessario incorporare caratteri di escape come mostrato nell'esempio seguente:

string columns = "Column 1\tColumn 2\tColumn 3";

Console.WriteLine(columns);

//Output: Column 1 Column 2 Column 3

string rows = "Row 1 \nRow 2 \nRow 3";

Console.WriteLine(rows);

/\* Output:

Row 1

Row 2

Row 3

\*/

**Sequenze di escape delle stringhe**

|  |  |
| --- | --- |
| **Sequenza di escape** | **Nome carattere** |
| \' | Virgoletta singola |
| \" | Virgoletta doppia |
| \\ | Barra rovesciata |
| \0 | Null |
| \n | Nuova riga |
| \t | Tabulazione orizzontale |

#### Simbolo @

Utilizzare stringhe verbatim per praticità e migliore leggibilità quando il testo della stringa contiene barre rovesciate, ad esempio nei percorsi di file. Poiché le stringhe verbatim mantengono i caratteri di nuova riga come parte del testo della stringa, possono essere utilizzate per inizializzare stringhe su più righe. Utilizzare virgolette doppie per incorporare una virgoletta in una stringa verbatim. Nell'esempio seguente sono mostrati alcuni utilizzi comuni delle stringhe verbatim:

string filePath = @"C:\Users\scoleridge\Documents\";

filePath -🡪 C:\Users\scoleridge\Documents\

string text = @"My pensive SARA ! thy soft cheek reclined

Thus on mine arm, most soothing sweet it is

To sit beside our Cot,...";

text -🡪 My pensive SARA ! thy soft cheek reclined

Thus on mine arm, most soothing sweet it is

To sit beside our Cot,...

string quote = @"Her name was ""Sara.""";

quote -🡪 Her name was "Sara."

#### Utilizzo di ToString()

Analogamente a tutti gli oggetti derivati da Object, le stringhe forniscono il metodo ToString, tramite il quale un valore viene convertito in una stringa. Questo metodo può essere utilizzato per convertire valori numerici in stringhe:

int year = 1999;

string msg = "Eve was born in " + year.ToString();

System.Console.WriteLine(msg); // outputs "Eve was born in 1999"

**Stringhe di formato**

Una stringa di formato è una stringa il cui contenuto può essere determinato dinamicamente in fase di esecuzione. Si crea una stringa di formato tramite il metodo Format statico e incorporando in parentesi graffe i segnaposto che verranno sostituiti da altri valori in fase di esecuzione. Nell'esempio seguente viene utilizzata una stringa di formato per restituire il risultato di ogni iterazione di un ciclo:

string s;

for (int i = 0; i < 10; i++)

{

// A simple format string with no alignment formatting.

s = String.Format("{0} times {1} = {2}", i, j, (i \* j));

Console.WriteLine(s);

}

**Stringhe null e stringhe vuote**

Una stringa vuota è un'istanza di un oggetto System.String che contiene zero caratteri. Le stringhe vuote vengono utilizzate spesso in diversi scenari di programmazione per rappresentare un campo di testo vuoto. È possibile chiamare metodi in stringhe vuote perché sono oggetti System.String validi. Le stringhe vuote sono inizializzate nel modo seguente:

string s = String.Empty;

Al contrario, una stringa null non fa riferimento a un'istanza di un oggetto System.String ed eventuali tentativi di chiamata di un metodo in una stringa null restituiscono un errore. È tuttavia possibile utilizzare stringhe null nelle operazioni di concatenazione e confronto con altre stringhe. Negli esempi seguenti vengono illustrati casi in cui un riferimento a una stringa null causa o meno la generazione di un'eccezione:

string str = "hello";

string nullStr = null;

string emptyStr = String.Empty;

string tempStr = str + nullStr;

// Output of the following line: hello

Console.WriteLine(tempStr);

bool b = (emptyStr == nullStr);

// Output of the following line: False

Console.WriteLine(b);

// The following line creates a new empty string.

string newStr = emptyStr + nullStr;

// Null strings and empty strings behave differently. The following two lines display 0.

Console.WriteLine(emptyStr.Length);

Console.WriteLine(newStr.Length);

// The following line raises a NullReferenceException.

//Console.WriteLine(nullStr.Length);

**Confronto tra caratteri singoli derivati da tipo string e il tipo char**

* **Carattere dichiarato di tipo char**

string StringaIniziale = "Stringa di prova", StringaFinale = "";

char Carattere = 'a';

for (int i = 0; i < StringaIniziale.Length; i++)

{

if (StringaIniziale[i] != Carattere)

StringaFinale = StringaFinale + StringaIniziale[i];

}

Console.WriteLine("Stringa Iniziale {0} ", StringaIniziale);

Console.WriteLine("Carattere da eliminare {0} ", Carattere);

Console.WriteLine("Stringa Finale {0} ", StringaFinale);

* **Carattere dichiarato di tipo string**

string StringaIniziale = "Stringa di prova", Carattere = "a", StringaFinale = "";

for (int i = 0; i < StringaIniziale.Length; i++)

{

*//if (StringaIniziale[i] != Carattere)* ***ERRORE***

if (StringaIniziale[i] != Carattere[0])

StringaFinale = StringaFinale + StringaIniziale[i];

}

Console.WriteLine("Stringa Iniziale {0} ", StringaIniziale);

Console.WriteLine("Carattere da eliminare {0} ", Carattere);

Console.WriteLine("Stringa Finale {0} ", StringaFinale);

* **Confronto con tipo char**

string StringaIniziale = "Stringa di prova", StringaFinale = "";

for (int i = 0; i < StringaIniziale.Length; i++)

{

if (StringaIniziale[i] != 'a')

StringaFinale = StringaFinale + StringaIniziale[i];

}

Console.WriteLine("Stringa Iniziale {0} ", StringaIniziale);

Console.WriteLine("Carattere da eliminare a ");

Console.WriteLine("Stringa Finale {0} ", StringaFinale);

* **Confronto con tipo char**

string StringaIniziale = "Stringa di prova", StringaFinale = "", a = "a";

for (int i = 0; i < StringaIniziale.Length; i++)

{

if (StringaIniziale[i] != a[0])

StringaFinale = StringaFinale + StringaIniziale[i];

}

Console.WriteLine("Stringa Iniziale {0} ", StringaIniziale);

Console.WriteLine("Carattere da eliminare a ");

Console.WriteLine("Stringa Finale {0} ", StringaFinale);

* **Confronto con tipo string**

string StringaIniziale = "Stringa di prova", StringaFinale = "";

for (int i = 0; i < StringaIniziale.Length; i++)

{

if (StringaIniziale[i] != "a"[0])

StringaFinale = StringaFinale + StringaIniziale[i];

}

Console.WriteLine("Stringa Iniziale {0} ", StringaIniziale);

Console.WriteLine("Carattere da eliminare a ");

Console.WriteLine("Stringa Finale {0} ", StringaFinale);

/\*

This code produces the following output.

Stringa Iniziale Stringa di prova

Carattere da eliminare a

Stringa Finale String di prov

\*/

**Membri String**

Rappresenta testo come una serie di caratteri Unicode.

# Costruttori pubblici

|  |  |
| --- | --- |
| **Nome** | **Descrizione** |
| String | Sottoposto a overload. Consente di inizializzare una nuova istanza della classe String. |

# Campi pubblici

|  |  |
| --- | --- |
| **Nome** | **Descrizione** |
| Empty | Rappresenta la stringa vuota. Il campo è in sola lettura. |

# Proprietà pubbliche

|  |  |
| --- | --- |
| **Nome** | **Descrizione** |
| Chars | Consente di ottenere il carattere alla posizione specificata in questa istanza. |
| Length | Consente di ottenere il numero di caratteri in questa istanza. |

**Proprietà String.Length**

La proprietà **Length** restituisce il numero di oggetti Char nell'istanza e non il numero dei caratteri Unicode. Un carattere Unicode, infatti, può essere rappresentato da più di un oggetto **Char**.

Nell'esempio di codice riportato di seguito viene illustrato come utilizzare la proprietà **Length**.

// Sample for String.Length

using System;

class Sample

{

public static void Main()

{

string str = "abcdefg";

Console.WriteLine("1) The length of '{0}' is {1}", str, str.Length);

Console.WriteLine("2) The length of '{0}' is {1}", "xyz", "xyz".Length);

}

}

/\*

This example produces the following results:

1) The length of 'abcdefg' is 7

2) The length of 'xyz' is 3

\*/

**Proprietà String.Chars**

La proprietà restituisce l'oggetto Char nella posizione specificata dal parametro index. Il parametro index è in base zero.

string miaStringa = "Stringa di prova";

for (int i = 0; i < miaStringa.Length; i++)

Console.Write("{0}-", miaStringa[i]);

Console.WriteLine();

/\*

This example produces the following results:

S-t-r-i-n-g-a- -d-i- -p-r-o-v-a-

\*/

# Metodi pubblici

|  |  |
| --- | --- |
| **Nome** | **Descrizione** |
| **Clone** | Consente di restituire un riferimento a questa istanza della classe **String** . |
| **Compare** | Sottoposto a overload. Consente di confrontare due oggetti **String** specificati. |
| **CompareTo** | Sottoposto a overload. Consente di eseguire il confronto fra l'istanza e un oggetto specificato o un oggetto **String** e restituisce un'indicazione dei valori relativi a essi associati. |
| **Contains** | Restituisce un valore che indica se l'oggetto **String** specificato è presente all'interno della stringa. |
| **CopyTo** | Consente di copiare un numero definito di caratteri da una posizione specificata in questa istanza in una posizione specificata in una matrice di caratteri Unicode. |
| **EndsWith** | Sottoposto a overload. Consente di determinare se la fine di un'istanza della classe **String** corrisponde a una stringa specificata. |
| **Format** | Sottoposto a overload. Consente di sostituire ogni elemento di formato presente in un oggetto **String** specificato con l'equivalente in forma di testo del valore di un oggetto corrispondente. |
| **IndexOf** | Sottoposto a overload. Restituisce l'indice della prima occorrenza di un oggetto **String**, oppure uno o più caratteri, all'interno di questa stringa. |
| **Insert** | Consente di inserire una determinata istanza dell'oggetto **String** alla posizione di indice specificata in questa istanza. |
| **IsNullOrEmpty** | Specifica se l'oggetto **String** indicato è riferimento null (**Nothing** in Visual Basic) o una stringa Empty. |
| **Join** | Consente di concatenare uno specifico oggetto **String** separatore a ciascun elemento di una matrice di oggetti String, producendo una stringa concatenata. |
| **LastIndexOf** | Sottoposto a overload. Restituisce la posizione di indice dell'ultima occorrenza di un carattere Unicode specificato o un oggetto **String** all'interno di questa istanza. |
| **PadLeft** | Sottoposto a overload. Allinea a destra i caratteri in questa istanza, aggiungendo a sinistra degli spazi o un determinato carattere Unicode fino a ottenere la lunghezza totale specificata. |
| **PadRight** | Sottoposto a overload. Allinea a sinistra i caratteri in questa stringa, aggiungendo a destra degli spazi o un determinato carattere Unicode fino a ottenere la lunghezza totale specificata. |
| **Remove** | Sottoposto a overload. Consente di eliminare un numero di caratteri specificato da questa istanza. |
| **Replace** | Sottoposto a overload. Consente di sostituire tutte le occorrenze di un determinato carattere Unicode o di un oggetto **String** presente in questa istanza con un altro carattere Unicode o un oggetto **String** specificato. |
| **Split** | Sottoposto a overload. Restituisce una matrice di oggetti **String** contenente le sottostringhe di questa istanza delimitate dagli elementi di una matrice di oggetti **Char** o **String** specificata. |
| **StartsWith** | Sottoposto a overload. Determina se l'inizio di un'istanza dell'oggetto **String** corrisponde a una stringa specificata. |
| **Substring** | Sottoposto a overload. Consente di recuperare una sottostringa da questa istanza. |
| **ToCharArray** | Sottoposto a overload. Consente di copiare i caratteri di questa istanza in una matrice di caratteri Unicode. |
| **ToLower** | Sottoposto a overload. Restituisce una copia di questo oggetto **String** convertito in caratteri minuscoli. |
| **ToString** | Sottoposto a overload. Sottoposto a override. Consente di convertire il valore di questa istanza in un oggetto **String**. |
| **ToUpper** | Sottoposto a overload. Restituisce una copia di questo oggetto **String** convertito in caratteri maiuscoli. |
| **Trim** | Sottoposto a overload. Consente di rimuovere tutte le occorrenze di una serie di caratteri specificati dalla parte iniziale e finale di questa istanza. |
| **TrimEnd** | Rimuove tutte le occorrenze di una serie di caratteri specificati in una matrice dalla parte finale di questa istanza. |
| **TrimStart** | Consente di rimuovere tutte le occorrenze di una serie di caratteri specificati in una matrice dalla parte iniziale di questa istanza. |

* **Metodo String.Compare (String, String)**

#### Valore restituito Valore integer con segno a 32 bit che consente di specificare la relazione lessicale tra i due termini di confronto.

|  |  |
| --- | --- |
| **Valore** | **Condizione** |
| Minore di zero | Il parametro strA è minore del parametro strB. |
| Zero | Il parametro strA è uguale al parametro strB. |
| Maggiore di zero | Il parametro strA è maggiore del parametro strB. |

* **Metodo String.Contains**

Restituisce un valore che indica se l'oggetto **String** specificato è presente all'interno della stringa: **true** se il parametro value è presente all'interno della stringa o se il parametro value è la stringa vuota (""); in caso contrario, **false**.

// This example demonstrates the String.Contains() method

using System;

class Sample

{

public static void Main()

{

string s1 = "The quick brown fox jumps over the lazy dog";

string s2 = "fox";

bool b;

b = s1.Contains(s2);

Console.WriteLine("Is the string, s2, in the string, s1?: {0}", b);

}

}

/\*

This example produces the following results:

Is the string, s2, in the string, s1?: True

\*/

* **Metodi String.EndsWith, String.StartsWith**

EndsWith: Consente di determinare se la fine di questa istanza di stringa corrisponde a una stringa specificata.

StartsWith: Consente di determinare se l’inizio di questa istanza di stringa corrisponde a una stringa specificata.

**Esempio**

String myString = "La mia stringa";

Console.WriteLine(myString.EndsWith("ga"));

Console.WriteLine(myString.StartsWith("Sa"));

Output:

True

False

* **Metodi String.IndexOf , String.LastIndexOf**

String.IndexOf : Restituisce l'indice in base zero della prima occorrenza della stringa specificata in questa istanza.

String.LastIndexOf : Restituisce l'indice in base zero dell’ultima occorrenza della stringa specificata in questa istanza.

* String.IndexOf (Char)

Restituisce l'indice della prima/ultima occorrenza del carattere Unicode specificato in questa stringa, oppure -1 in caso contrario.

* String.IndexOf (String)

Restituisce l'indice della prima/ultima occorrenza dell'oggetto **String** specificato in questa istanza, oppure -1 in caso contrario.

* String.IndexOf (Char, Int32)

Restituisce l'indice della prima/ultima occorrenza del carattere Unicode specificato in questa stringa. La ricerca ha inizio dalla posizione del carattere specificata, oppure -1 in caso contrario.

* String.IndexOf (String, Int32)

Restituisce l'indice della prima/ultima occorrenza dell'oggetto **String** specificato in questa istanza. La ricerca ha inizio alla posizione del carattere specificata, oppure -1 in caso contrario.

* String.IndexOf (Char, Int32, Int32)

Restituisce l'indice della prima/ultima occorrenza del carattere specificato in questa istanza. La ricerca ha inizio dalla posizione specificata dei caratteri ed esamina un determinato numero di posizioni.

* String.IndexOf (String, Int32, Int32)

Restituisce l'indice della prima/ultima occorrenza dell'oggetto String specificato in questa istanza. La ricerca viene eseguita iniziando dalla posizione specificata dei caratteri e esamina un determinato numero di posizioni.

**Esempio**

string Stringa = "012345678901234";

int Posizione = Stringa.IndexOf("2");

Console.WriteLine(Posizione);

// Output ---> 2

//

Posizione = Stringa.IndexOf("2", 5);

Console.WriteLine(Posizione);

// Output ---> 12

//

Posizione = Stringa.IndexOf("2", 5, 4);

Console.WriteLine(Posizione);

// Output ---> -1

//

Posizione = Stringa.IndexOf("NO");

Console.WriteLine(Posizione);

// Output ---> -1

//

Posizione = Stringa.LastIndexOf("2");

Console.WriteLine(Posizione);

// Output ---> 12

* **Metodo String.Insert**

Consente di inserire una determinata istanza dell'oggetto **String** alla posizione di indice specificata in questa istanza.

returnValue = instance.Insert(startIndex, value)

**Esempio**

string miaStringa = "Stringa di prova";

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Remove(7));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Remove(7,3));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Insert(8, "è "));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Replace("p","t"));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Replace('p', 't'));

/\*

La stringa -Stringa di prova- diventa -Stringa-

La stringa -Stringa di prova- diventa -Stringa prova-

La stringa -Stringa di prova- diventa -Stringa è di prova-

La stringa -Stringa di prova- diventa -Stringa di trova-

La stringa -Stringa di prova- diventa -Stringa di trova-

\*/

* **Metodi String.PadLeft, String.PadRight**

Allinea a destra/sinistra i caratteri in questa istanza, aggiungendo a sinistra/destra degli spazi o un determinato carattere Unicode fino a ottenere la lunghezza totale specificata.

* String.PadLeft (Int32)

Allinea a destra i caratteri in questa istanza, aggiungendo a sinistra degli spazi fino a ottenere la lunghezza totale specificata.

* String.PadLeft (Int32, Char)

Allinea a destra i caratteri in questa istanza, aggiungendo a sinistra un determinato carattere Unicode fino a ottenere la lunghezza specificata.

string miaStringa = "Stringa di prova";

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.PadLeft(30));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.PadRight(30,'\*'));

Console.WriteLine();

/\*

La stringa -Stringa di prova- diventa - Stringa di prova-

La stringa -Stringa di prova- diventa -Stringa di prova\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*/

* **Metodo String.Remove**

Consente di eliminare un numero di caratteri specificato da questa istanza.

* String.Remove (Int32)

Consente di eliminare tutti i caratteri dalla stringa a partire da una posizione specifica e continuando fino all'ultima posizione.

* String.Remove (Int32, Int32)

Consente di eliminare un determinato numero di caratteri da questa istanza, a partire dalla posizione specificata.

returnValue = instance.Remove(startIndex, count)

**Esempio**

string miaStringa = "Stringa di prova";

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Remove(7));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Remove(7,3));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Insert(8, "è "));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Replace("p","t"));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Replace('p', 't'));

/\*

La stringa -Stringa di prova- diventa -Stringa-

La stringa -Stringa di prova- diventa -Stringa prova-

La stringa -Stringa di prova- diventa -Stringa è di prova-

La stringa -Stringa di prova- diventa -Stringa di trova-

La stringa -Stringa di prova- diventa -Stringa di trova-

\*/

* **Metodo String.Replace**

Consente di sostituire tutte le occorrenze di un determinato carattere Unicode o di un oggetto String presente in questa istanza con un altro carattere Unicode o un oggetto String specificato.

returnValue = instance.Replace(oldValue, newValue)

**Esempio**

string miaStringa = "Stringa di prova";

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Remove(7));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Remove(7,3));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Insert(8, "è "));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Replace("p","t"));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.Replace('p', 't'));

/\*

La stringa -Stringa di prova- diventa -Stringa-

La stringa -Stringa di prova- diventa -Stringa prova-

La stringa -Stringa di prova- diventa -Stringa è di prova-

La stringa -Stringa di prova- diventa -Stringa di trova-

La stringa -Stringa di prova- diventa -Stringa di trova-

\*/

* **Metodo String.Substring**

Consente di recuperare una sottostringa da questa istanza.

* String.Substring (Int32)

Consente di recuperare una sottostringa da questa istanza. La sottostringa ha inizio alla posizione del carattere specificata.

* String.Substring (Int32, Int32)

Consente di recuperare una sottostringa da questa istanza. La sottostringa ha inizio alla posizione del carattere specificata e ha una determinata lunghezza.

returnValue = instance.Substring(startIndex, length)

**Esempio**

String myString = "abc";

bool test1 = String.Compare(myString.Substring(2, 1), "c") == 0; // This is true.

myString.Substring(3, 1); // This throws ArgumentOutOfRangeException.

bool test2 = String.Compare(myString.Substring(3, 0), String.Empty) == 0; // This is true.

* **Metodi String.ToUpper e String.ToLower**
* String.ToUpper ()

Restituisce una copia dell'oggetto String convertito in caratteri maiuscoli, utilizzando le regole relative all'utilizzo di maiuscole e minuscole proprie della lingua corrente.

* String.ToLower ()

Restituisce una copia dell'oggetto String convertito in caratteri minuscoli, utilizzando le regole relative all'utilizzo di maiuscole e minuscole proprie della lingua corrente.

**Esempio**

using System;

public class ToLowerTest {

public static void Main() {

string [] info = {"Name", "Title", "Age", "Location", "Gender"};

Console.WriteLine("The initial values in the array are:");

foreach (string s in info)

Console.WriteLine(s);

Console.WriteLine("{0}The lowercase of these values is:", Environment.NewLine);

foreach (string s in info)

Console.WriteLine(s.ToLower());

Console.WriteLine("{0}The uppercase of these values is:", Environment.NewLine);

foreach (string s in info)

Console.WriteLine(s.ToUpper());

}

}

* **Metodi String.TrimStart, String.TrimEnd**

Consente di rimuovere tutte le occorrenze di una serie di caratteri specificati in una matrice dalla parte iniziale/finale di questa istanza.

**Parametri**

trimChars: Matrice di caratteri Unicode da rimuovere oppure riferimento null

**Valore restituito**

Oggetto String che si ottiene dopo la rimozione di tutte le occorrenze dei caratteri nel parametro trimChars dalla parte iniziale. Se il parametro trimChars è riferimento null (Nothing in Visual Basic), verranno rimossi invece gli spazi vuoti.

string miaStringa = "Stringa di prova ";

char[] MatriceCar = new char[] { 'S', 't' };

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.TrimStart(MatriceCar));

Console.WriteLine("La stringa -{0}- diventa -{1}-", miaStringa, miaStringa.TrimEnd());

Console.WriteLine();

/\*

La stringa -Stringa di prova - diventa -ringa di prova -

La stringa -Stringa di prova - diventa -Stringa di prova-

\*/

* **Metodo String.Trim**

Consente di rimuovere tutte le occorrenze di una serie di caratteri specificati dalla parte iniziale e finale di questa istanza.

* String.Trim()

Consente di rimuovere tutte le occorrenze di caratteri spazi vuoti dalla parte iniziale e finale di questa istanza.

* String.Trim(Char[])

Consente di rimuovere tutte le occorrenze di una serie di caratteri specificati in una matrice dalla parte iniziale e finale di questa istanza.

**Metodo String.Split**

Restituisce una matrice di oggetti String contenente le sottostringhe di questa istanza delimitate dagli elementi di una matrice di oggetti Char o String specificata.

**Esempio**

string stringa = "Cane,Gatto,Tigre,Leone,Asino";

string[] parole = stringa.Split(',');

foreach (string parola in parole)

{

Console.WriteLine(parola);

}

// Output:

Cane

Gatto

Tigre

Leone

Asino

**Esempio**

string stringa = "Cane,Gatto;Tigre,Leone:Asino";

string[] parole = stringa.Split(new char[] { ',', ';', ':' });

foreach (string parola in parole)

{

Console.WriteLine(parola);

}

// Output:

Cane

Gatto

Tigre

Leone

Asino

* **Metodo String.Join**

Consente di concatenare uno specifico oggetto String separatore a ciascun elemento di una determinata matrice di oggetti String, producendo una singola stringa concatenata.

returnValue = String.Join(separator, value)

**Esempio**

string[] parole = { "Cane", "Gatto", "Tigre", "Leone", "Asino" };

Console.WriteLine(String.Join("---", parole));

Console.WriteLine();

Console.WriteLine(String.Join("---", parole, 0 , 3));

// Output:

Cane---Gatto---Tigre---Leone---Asino

Cane---Gatto---Tigre

* **Metodo String.CopyTo**

Consente di copiare un numero definito di caratteri da una posizione specificata in questa istanza in una posizione specificata in una matrice di caratteri Unicode.

Parametri:

sourceIndex: Posizione del carattere in questa istanza.

Destination: Matrice di caratteri Unicode.

destinationIndex: Elemento della matrice nel parametro destination.

count

Numero di caratteri presenti in questa istanza da copiare nel parametro destination.

**Esempio**

using System;

public class CopyToTest

{

public static void Main()

{

// Embed an array of characters in a string

string strSource = "changed";

char[] destination = { 'T', 'h', 'e', ' ', 'i', 'n', 'i', 't', 'i', 'a', 'l', ' ',

'a', 'r', 'r', 'a', 'y' };

// Print the char array

Console.WriteLine(destination);

// Embed the source string in the destination string

strSource.CopyTo(0, destination, 4, strSource.Length);

// Print the resulting array

Console.WriteLine(destination);

strSource = "A different string";

// Embed only a section of the source string in the destination

strSource.CopyTo(2, destination, 3, 9);

// Print the resulting array

Console.WriteLine(destination);

Console.ReadKey();

}

}

// Output:

The initial array

The changed array

Thedifferentarray

* **Metodo String.ToCharArray**

Consente di copiare i caratteri di questa istanza in una matrice di caratteri Unicode.

String.ToCharArray () Consente di copiare i caratteri di questa istanza in una matrice di caratteri Unicode.

String.ToCharArray (Int32, Int32) Consente di copiare i caratteri di una determinata sottostringa di questa istanza in una matrice di caratteri Unicode.

**Esempio**

// Sample for String.ToCharArray(Int32, Int32)

using System;

class Sample {

public static void Main() {

string str = "012wxyz789";

char[] arr;

arr = str.ToCharArray(3, 4);

Console.Write("The letters in '{0}' are: '", str);

Console.Write(arr);

Console.WriteLine("'");

Console.WriteLine("Each letter in '{0}' is:", str);

foreach (char c in arr)

Console.WriteLine(c);

}

}

/\*

This example produces the following results:

The letters in '012wxyz789' are: 'wxyz'

Each letter in '012wxyz789' is:

w

x

y

z

\*/