Academy .NET

Stringhe e Text Processing

Sommario

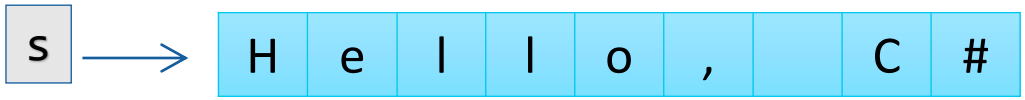
- Creazione e utilizzo di stringhe
- Manipolazione di stringhe
- Altre operazioni su stringhe
- Creazione e modifica di stringhe
- Formattazione delle stringhe

Le stringhe

- Le stringhe sono sequenze di caratteri
- Ogni carattere è un simbolo **Unicode** rappresentato dal tipo di dati stringa in C# (**System.String**)

- Esempio:

```
string s = "Hello, C#";
```



Classe System.String

- Le stringhe in .NET Framework sono rappresentate da oggetti **System.String**

  Gli oggetti stringa contengono una sequenza di caratteri **immutabili** (di sola lettura)

- Le stringhe vengono archiviate nella memoria dinamica (**heap memory**)
- **System.String è un tipo riferimento**

Classe System.String

- Gli oggetti stringa sono come matrici di caratteri (**char[]**)

  Hanno una lunghezza fissa (**String.Length**)
  È possibile accedere agli elementi direttamente dall'indice
  L'indice è compreso nell'intervallo [0 ... Length-1]

```
string s = "Hello!";
int len = s.Length; // len = 6
char ch = s[1]; // ch = 'e'
```

index =	0	1	2	3	4	5
s[index] =	H	e	l	l	o	!

Stringhe - Esempio

```
static void Main()
{
    string s =
        "Stand up, stand up, Balkan Superman.";
    Console.WriteLine("s = \"{0}\"", s);
    Console.WriteLine("s.Length = {0}", s.Length);
    for (int i = 0; i < s.Length; i++)
    {
        Console.WriteLine("s[{0}] = {1}", i, s[i]);
    }
}
```

Dichiarazione di una stringa

Diversi modi per dichiarare variabili stringa:

- Utilizzando la parola chiave **string**
- Utilizzando il nome di classe completo **System.String**

```
string str1;
System.String str2;
String str3;
```

NOTA: Le tre dichiarazioni precedenti sono equivalenti

Creazione di stringhe

Prima di essere inizializzata, una variabile **string** ha valore **null.**

- Le stringhe possono essere inizializzate da:
  Assegnazione di una stringa alla variabile stringa.
  Assegnazione del valore di un'altra variabile stringa.
  Assegnazione del risultato di una operazione di tipo stringa.

Creazione di stringhe

Le variabili non inizializzate hanno valore **null.**

```
string s;
```

Assegnazione di una stringa.

```
string s = "I am a string literal!";
```

Assegnazione da un'altra variabile di stringa.

```
string s2 = s;
```

Conversione del risultato dell'operazione su stringa.

```
string s = 42.ToString();
```

Lettura e Scrittura di stringhe

Lettura di stringhe dalla console:

Utilizzo del metodo **Console.ReadLine()**

```
string s = Console.ReadLine();
```

Stampa di stringhe sulla console:

Utilizzo dei metodi **Write()** e **WriteLine()**

```
Console.Write("Please enter your name: ");
string name = Console.ReadLine();
Console.Write("Hello, {0}! ", name);
Console.WriteLine("Welcome to our party!");
```

Comparazione di stringhe

Diversi modi per confrontare due stringhe:

Confronto di stringhe **dictionary-based**

Case-insensitive

```
int result = string.Compare(str1, str2, true);
// result == 0 if str1 equals str2
// result < 0 if str1 is before str2
// result > 0 if str1 is after str2
```

Case-sensitive

```
string.Compare(str1, str2, false);
```

Comparazione di stringhe

- Controllo di uguaglianza con operatore **==**

  Esegue il confronto con distinzione tra maiuscole e minuscole

```
if (str1 == str2)
{
    …
}
```

- Utilizzando il metodo **Equals()** con distinzione tra maiuscole e minuscole

  Lo stesso effetto dell'operatore **==**

```
if (str1.Equals(str2))
{
    …
}
```

Concatenazione di stringhe

Esistono due modi per combinare le stringhe:

Utilizzando il metodo **Concat()**

```
string str = String.Concat(str1, str2);
```

Utilizzando gli operatori **+** o **+=**

```
string str = str1 + str2 + str3;
string str += str1;
```

- Qualsiasi oggetto può essere aggiunto ad una stringa:

```
string name = "Peter";
int age = 22;
string s = name + " " + age; //   "Peter 22"
```

Concatenazione di stringhe - Esempio

```
string firstName = "Telerik";
string lastName = "Academy";

string fullName = firstName + " " + lastName;
Console.WriteLine(fullName);
// Telerik Academy

int age = 5;

string nameAndAge =
    "Name: " + fullName +
    "\nAge: " + age;
Console.WriteLine(nameAndAge);
// Name: Telerik Academy
// Age: 5
```

IndexOf(...)

Trovare un carattere o una sottostringa all'interno di una determinata stringa:

Indice di una stringa

```
IndexOf(string str)
```

Indice di una stringa a partire da una data posizione

```
IndexOf(string str, int startIndex)
```

Ultimo indice

```
LastIndexOf(string)
```

IndexOf(...) - Esempio

```
string str = "C# Programming Course";
int index = str.IndexOf("C#"); // index = 0
index = str.IndexOf("Course"); // index = 15
index = str.IndexOf("COURSE"); // index = -1
// IndexOf is case-sensetive. -1 means not found
index = str.IndexOf("ram"); // index = 7
index = str.IndexOf("r"); // index = 4
index = str.IndexOf("r", 5); // index = 7
index = str.IndexOf("r", 8); // index = 18
```

index =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	…
s[index] =	C	#		P	r	o	g	r	a	m	m	i	n	g	…

Estrazione di sottostringhe

str.Substring (int startIndex, int length)

```
string filename = @"C:\Pics\Rila2009.jpg";
string name = filename.Substring(8, 8);
// name is Rila2009
```

str.Substring (int startIndex)

```
string filename = @"C:\Pics\Summer2009.jpg";
string nameAndExtension = filename.Substring(8);
// nameAndExtension is Summer2009.jpg
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C	:	\	P	i	c	s	\	R	i	l	a	2	0	0	5	.	j	p	g

Dividere stringhe

- Per dividere una stringa in base a uno o più separatori, utilizzare il metodo seguente: `string[] Split(params char[])`

- Esempio:

```
string listOfBeers =
    "Amstel, Zagorka, Tuborg, Becks.";
string[] beers =
    listOfBeers.Split(' ', ',', '.');
Console.WriteLine("Available beers are:");
foreach (string beer in beers)
{
    Console.WriteLine(beer);
}
```

Altre operazioni con le stringhe

- **Replace(string, string)**: sostituisce tutte le occorrenze di una determinata stringa con un'altra

  Il risultato è una nuova stringa (le stringhe non sono modificabili)

```
string cocktail = "Vodka + Martini + Cherry";
string replaced = cocktail.Replace("+", "and");
// Vodka and Martini and Cherry
```

- **Remove(index, length):** Restituisce una nuova stringa in cui è stato eliminato un numero specificato di caratteri nell'istanza corrente a partire da una posizione specificata.

```
string price = "$ 1234567";
string lowPrice = price.Remove(2, 3);
// $ 4567
```

Altre operazioni con le stringhe

- Da maiuscolo a minuscolo e viceversa:

Utilizzo del metodo **ToLower()**

```
string alpha = "aBcDeFg";
string lowerAlpha = alpha.ToLower(); // abcdefg
Console.WriteLine(lowerAlpha);
```

Utilizzo del metodo **ToUpper()**

```
string alpha = "aBcDeFg";
string upperAlpha = alpha.ToUpper(); // ABCDEFG
Console.WriteLine(upperAlpha);
```

Utilizzo di Trim()

- Utilizzo di **Trim()**

```
string s = "    example of white space    ";
string clean = s.Trim();
Console.WriteLine(clean);
```

- Utilizzo di **Trim(chars)**

```
string s = " \t\nHello!!! \n";
string clean = s.Trim(' ', ',' ,'!', '\n','\t');
Console.WriteLine(clean); // Hello
```

- Utilizzo di **TrimStart()** e **TrimEnd()**

```
string s = "   C#   ";
string clean = s.TrimStart(); // clean = "C#   "
```

Immutabilità delle stringhe

- Le stringhe sono **immutabili**!

  Concat(), **Replace()**, **Trim()**, ... restituiscono una nuova stringa, non la modificano.

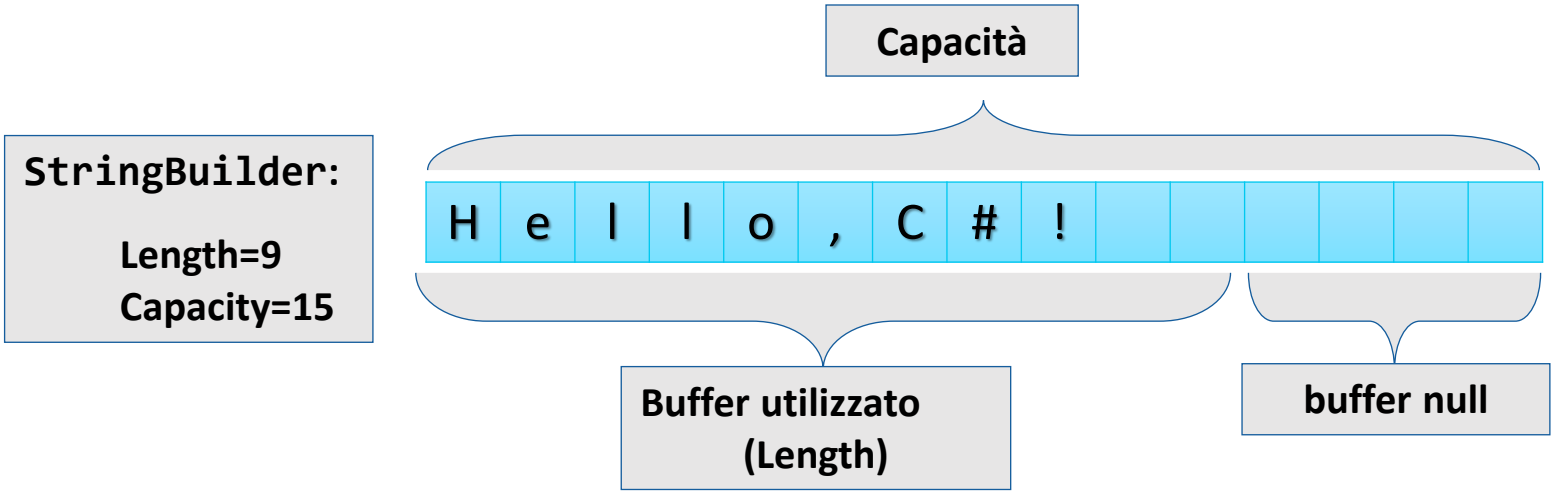
- Evitare l'utilizzo di "**+**" per le stringhe in un ciclo

  Funziona in modo molto inefficiente

```
public static string DupChar(char ch, int count)
{
    string result = "";
    for (int i=0; i<count; i++)
        result += ch;
    return result;
}
```

Pratica non corretta. Da evitare

StringBuilder



StringBuilder mantiene una **buffer memory**, allocata in anticipo.

La maggior parte delle operazioni utilizza la memoria buffer e non alloca nuovi oggetti.

Stringhe e operatore +

- Considerare la seguente concatenazione di stringhe:

```
string result = str1 + str2;
```

- È equivalente al seguente codice:

```
StringBuilder sb = new StringBuilder();
sb.Append(str1);
sb.Append(str2);
string result = sb.ToString();
```

- Nuovi oggetti vengono creati e mantenuti dal **Garbage Collector** per la deallocazione

Classe StringBuilder

- Il costruttore **StringBuilder(int** capacità**)** alloca in anticipo il buffer di una data dimensione

  Per impostazione predefinita, vengono assegnati 16 caratteri

- **Capacity** contiene lo spazio attualmente allocato (in caratteri)

- **this[int index]** (indexer in C#) dà accesso al valore **char** in una data posizione

- **Length** mantiene la lunghezza della stringa nel buffer

Classe StringBuilder

- **Append(…)** aggiunge una stringa o un altro oggetto dopo l'ultimo carattere nel buffer
- **Remove(int startIndex, int length)** rimuove i caratteri nell'intervallo dato
- **Insert(int index, string str)** inserisce una data stringa (o oggetto) in una data posizione
- **Replace(string oldStr, string newStr)** sostituisce tutte le occorrenze di una sottostringa
- **ToString()** converte StringBuilder in String

Modificare il contenuto con StringBuilder

- Utilizzo della classe **System.Text.StringBuilder** per stringhe di caratteri modificabili:

```
public static string ReverseString(string s)
{
    StringBuilder sb = new StringBuilder();
    for (int i = s.Length-1; i >= 0; i--)
        sb.Append(s[i]);
    return sb.ToString();
}
```

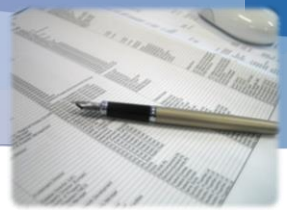
- Utilizza **StringBuilder** se è necessario continuare ad aggiungere caratteri in una stringa

StringBuilder - Esempio

- Estrazione di tutte le lettere maiuscole da una stringa:

```
public static string ExtractCapitals(string s)
{
    StringBuilder result = new StringBuilder();
    for (int i = 0; i<s.Length; i++)
    {
      if (Char.IsUpper(s[i]))
        {
            result.Append(s[i]);
        }
    }
    return result.ToString();
}
```

Formattazione di stringhe



Tutte le classi in C# hanno un metodo virtuale pubblico **ToString():**

Restituisce una stringa leggibile e culture-sensitive che rappresenta l'oggetto

La maggior parte dei tipi di .NET Framework ha una propria implementazione di ToString():

int, **float**, **bool**, **DateTime**

```
int number = 5;
string s = "The number is " + number.ToString();
Console.WriteLine(s); // The number is 5
```

Metodo String.Format()

- Applica modelli per la formattazione delle stringhe
  I segnaposto vengono utilizzati per il testo dinamico
  Come **Console.WriteLine(...)**

```
string template = "If I were {0}, I would {1}.";
string sentence1 = String.Format(
    template, "developer", "know C#");
Console.WriteLine(sentence1);
// If I were developer, I would know C#.

string sentence2 = String.Format(
    template, "elephant", "weigh 4500 kg");
Console.WriteLine(sentence2);
// If I were elephant, I would weigh 4500 kg.
```

Formattazione di date

- Le date hanno le proprie stringhe di formattazione:

d, **dd** - giorno (con / senza zero iniziale)
M, **MM** - mese
yy, **yyyy** - anno (2 o 4 cifre)
h, **HH**, **m**, **mm**, **s**, **ss** - ora, minuti, secondi

```
DateTime now = DateTime.Now;
Console.WriteLine(
    "Now is {0:d.MM.yyyy HH:mm:ss}", now);
// Now is 31.11.2009 11:30:32
```

Domande & approfondimenti

Academy .NET