BCSCHOOL

Empowering Digital Skills For The Jobs Of The Future

by

BC Soft
consulenza sistemi informativi

AiRED
Artificial Intelligence Ricerca e Sviluppo

Academy .NET

Dizionari

Sommario

- I dizionari
- Dictionary<K, V> vs. SortedDictionary<K, V>
- Dizionari: Funzionalità
- **LINQ**
  - Cos'è LINQ?
  - Operazioni matematiche con LINQ
  - Leggere le collezioni in una sola riga

I Dizionari

- I dizionari sono array indicizzati da chiavi
- Sono formati da una **coppia** {**chiave -> valore**}

Array

key	**0**	**1**	**2**	**3**	**4**
value	8	-3	12	408	33

Dizionario

key	**value**
John Smith	+1-555-8976
Lisa Smith	+1-555+1234
Sam Doe	+1-555-5030

Dizionario: Esempio - Phonebook

```
var phonebook = new Dictionary<string, string>();
phonebook["John Smith"] = "+1-555-8976";
phonebook["Lisa Smith"] = "+1-555-1234";
phonebook["Sam Doe"] = "+1-555-5030";
phonebook["Nakov"] = "+359-899-555-592";
phonebook["Nakov"] = "+359-2-981-9819"; // Replace
phonebook.Remove("John Smith");
foreach (var pair in phonebook)
  Console.WriteLine("{0} --> {1}",
    pair.Key, pair.Value);
```

Dictionary<K, V> vs. SortedDictionary<K, V>

- **Dizionario**

  Utilizza una tabella **hash + list**

  Dictionary <K, V>

  Conserva le chiavi nell'ordine in cui sono state aggiunte

```
var dict =
  new Dictionary<string, int>();
```

- **Dizionario ordinato**

  Utilizza un **balanced search tree**

  SortedDictionary <K, V>

  Mantiene le chiavi ordinate nel loro ordine naturale

```
var sortedDict = new
  SortedDictionary<int,int>();
```

Dizionari: Funzionalità

- **Count**: contiene il numero di coppie chiave-valore
- **Keys**: un set di chiavi univoche

```
var dict = new Dictionary<string, int>();
foreach(var key in dict.Keys)
  Console.WriteLine(key);
```

- **Values**: una raccolta di tutti i valori

```
Console.WriteLine(String.Join(", ", dict.Values));
```

- Operazioni di base: **Add()** / ,**Remove()**, **Clear()**

Dizionari: Funzionalità

- Metodi per la ricerca di **chiave**/**valore**:

ContainsKey() - controlla se una **chiave** è presente nel dizionario

ContainsValue() - controlla se un **valore** è presente nel dizionario

TryGetValue(): controlla se una chiave è presente nel dizionario e restituisce il valore(o restituisce il valore di default)

SortedDictionary Esempio - Events

```
var events = new SortedDictionary<DateTime, string>();
events[new DateTime(1998, 9, 4)] = "Google's birth date";
events[new DateTime(2013, 11, 5)] = "SoftUni's birth date";
events[new DateTime(1975, 4, 4)] = "Microsoft's birth date";
events[new DateTime(2004, 2, 4)] = "Facebook's birth date";
events[new DateTime(2013, 11, 5)] = "SoftUni was founded";
foreach (var entry in events)
{
   Console.WriteLine("{0:dd-MMM-yyyy}: {1}",
      entry.Key, entry.Value);
```

Cosa è LINQ?

LINQ = **L**anguage **I**ntegrated **Q**uery

Semplifica notevolmente il collection processing

```
List<int> nums = new List<int>() { 2, 4, 3, 1 };
var sum = nums.Sum(); // 10
var min = nums.Min(); // 1
var avg = nums.Average(); // 2.5
var bigNums = nums.Where(a => a > 2).ToList(); // {4, 3}
var reversed = nums.Reverse().ToList(); // { 1, 3, 4, 2 }
```

Processing Sequences con LINQ

- **Min()**: trova l'elemento più piccolo in una raccolta

```
new List<int>() { 1, 2, 3, 4, -1, -5, 0, 50 }.Min()     return -5
```

- **Max()**: trova l'elemento più grande in una raccolta

```
new int[] { 1, 2, 3, 40, -1, -5, 0, 5 }.Max()           return 40
```

- **Sum()**: esegue la somma di tutti gli elementi in una raccolta

```
new long[] {1, 2, 3, 4, -1, -5, 0, 50}.Sum()            return  54
```

- **Average()**: calcola la media degli elementi in una raccolta

```
new int[] {1, 2, 3, 4, -1, -5, 0, 50}.Average()         return 6.75
```

Leggere collezioni su singola riga

Utilizzo di **Select()** per leggere le collezioni:

```
   var nums = Console.ReadLine()
   .Split()
   .Select(number => double.Parse(number));
   // .Select(double.Parse);
```

```
var nums = Console.ReadLine()
   .Split()
   .Select(int.Parse);
// .Select(number => int.Parse(number));
```

Convertire Collezioni

Utilizzare **ToArray()**, **ToList()** per convertire le collezioni:

```
int[] nums = Console.ReadLine()
        .Split()
        .Select(number => int.Parse(number))
        .ToArray();
```

```
List<double> nums = Console.ReadLine()
        .Split()
        .Select(double.Parse)
        .ToList();
```

Ordinare Collezioni

Utilizzo di **OrderBy()**:

```
List<int> nums = { 1, 5, 2, 4, 3 };
nums = nums
        .OrderBy(num => num)
        .ToList()
```

Utilizzo di **OrderByDescending()**:

```
List<int> nums = { 1, 5, 2, 4, 3 };
nums = nums
        .OrderByDescending(num => num)
        .ToList()
```

Ordinare collezioni con criteri multipli

Utilizzo di **ThenBy()** per ordinare con più di un criterio:

```
Dictionary<int, string> products =
  new Dictionary<int, string>();
Dictionary<int, string> sortedDict = products
  .OrderBy(pair => pair.Value)
  .ThenBy(pair => pair.Key)
  .ToDictionary(pair => pair.Key, pair => pair.Value);
```

Utilizzo di Take() e Skip() nelle collezioni

Utilizzare **Take()** e **Skip()**:

```
var nums = new List<int>() { 10, 20, 30, 40, 50, 60}
  .Take(3)
  .ToArray();
// nums = [10, 20, 30]
```

```
var nums = new List<int>() { 10, 20, 30, 40, 50, 60}
  .Skip(3).Take(2)
  .ToArray();
// nums = [40, 50]
```

Filtrare collezioni

- Utilizzo di **Where()** e **Count()**:

```
int[] nums = { 1, 2, 3, 4, 5, 6};
nums = nums
     .Where(num => num % 2 == 0)
     .ToArray();
// nums = [2, 4, 6]
```

```
int[] nums = { 1, 2, 3, 4, 5, 6};
int count = nums.Count(num => num % 2 == 0);
// count = 3
```

Filtrare e Ordinare con Espressioni Lambda

```
int[] nums = { 11, 99, 33, 55, 77, 44, 66, 22, 88 };

nums.OrderBy(x => x).Take(3);
// 11 22 33
nums.Where(x => x < 50);
// 11 33 44 22
nums.Count(x => x % 2 == 1);
// 5
nums.Select(x => x * 2).Take(5);
// 22 198 66 110 154
```

Evitare duplicazioni in una Collezione

Si può utilizzare **Distinct()** per ottenere solamente elementi distinti in una collezione:

```
int[] nums =
{ 1, 2, 2, 3, 4, 5, 6, -2, 2, 0, 15, 3, 1, 0, 6 };

nums = nums
    .Distinct()
    .ToArray();
// nums = [1, 2, 3, 4, 5, 6, -2, 0, 15]
```

Domande & approfondimenti

Academy .NET