# static

http://msdn.microsoft.com/it-it/library/98f28cdx.aspx

Utilizzare il modificatore static per dichiarare un membro statico che appartiene al tipo stesso anziché a un oggetto specifico. Il modificatore static può essere utilizzato con classi, campi, metodi, proprietà, operatori, eventi e costruttori, ma non con indicizzatori, distruttori o tipi che non sono classi. Per ulteriori informazioni, vedere Classi statiche e membri di classi statiche.

**Esempio:** La classe seguente viene dichiarata come static e contiene soltanto metodi static:

static class CompanyEmployee

{

public static void DoSomething() { /\*...\*/ }

public static void DoSomethingElse() { /\*...\*/ }

}

Una dichiarazione di costanti o di tipi è in modo implicito un membro static.

Non è possibile fare riferimento a un membro static mediante un'istanza. È invece possibile ottenere il risultato desiderato utilizzando il nome del tipo in cui è dichiarato. Si consideri ad esempio la seguente classe:

public class MyBaseC

{

public struct MyStruct

{

public static int x = 100;

}

}

Per fare riferimento al membro statico x, utilizzare il nome completo, MyBaseC.MyStruct.x, a meno che il membro non sia accessibile dallo stesso ambito:

Console.WriteLine(MyBaseC.MyStruct.x);

Mentre l'istanza di una classe contiene una copia distinta di tutti i campi di istanza della classe, esiste una sola copia per ciascun campo static.

Non è possibile utilizzare this per fare riferimento a metodi statici o a funzioni di accesso della proprietà statiche.

Se la parola chiave static viene applicata a una classe, tutti i membri della classe devono essere statici.

Le classi e le classi statiche potrebbero avere costruttori statici. I costruttori statici vengono chiamati in un determinato momento compreso tra l'avvio del programma e la creazione di un'istanza della classe.

Per illustrare le caratteristiche dei membri statici, si consideri una classe che rappresenta un dipendente di un'azienda. Si supponga che la classe contenga un metodo per conteggiare i dipendenti e un campo per memorizzarne il numero. Sia il metodo che il campo non appartengono ad alcuna istanza di tipo dipendente, in quanto appartengono alla classe azienda e devono pertanto essere dichiarati come membri static della classe.

In questo esempio si leggono il nome e l'identificatore di un nuovo dipendente; il contatore di dipendenti viene quindi incrementato di un'unità e vengono visualizzate le informazioni relative al nuovo dipendente e al numero di dipendenti aggiornato. Per semplicità, questo programma legge il numero attuale di dipendenti dalla tastiera. In un'applicazione reale, queste informazioni verrebbero lette da un file.

public class Employee4

{

public string id;

public string name;

public Employee4()

{

}

public Employee4(string name, string id)

{

this.name = name;

this.id = id;

}

public static int employeeCounter;

public static int AddEmployee()

{

return ++employeeCounter;

}

}

class MainClass : Employee4

{

static void Main()

{

Console.Write("Enter the employee's name: ");

string name = Console.ReadLine();

Console.Write("Enter the employee's ID: ");

string id = Console.ReadLine();

// Create and configure the employee object:

Employee4 e = new Employee4(name, id);

Console.Write("Enter the current number of employees: ");

string n = Console.ReadLine();

Employee4.employeeCounter = Int32.Parse(n);

Employee4.AddEmployee();

// Display the new information:

Console.WriteLine("Name: {0}", e.name);

Console.WriteLine("ID: {0}", e.id);

Console.WriteLine("New Number of Employees: {0}",

Employee4.employeeCounter);

}

}

/\*

Input:

Matthias Berndt

AF643G

15

\*

Sample Output:

Enter the employee's name: Matthias Berndt

Enter the employee's ID: AF643G

Enter the current number of employees: 15

Name: Matthias Berndt

ID: AF643G

New Number of Employees: 16

\*/

In questo esempio viene mostrato che, sebbene sia possibile inizializzare un campo statico tramite un altro campo statico non ancora dichiarato, i risultati non saranno definiti finché non verrà assegnato un valore al campo statico in modo esplicito.

class Test

{

static int x = y;

static int y = 5;

static void Main()

{

Console.WriteLine(Test.x);

Console.WriteLine(Test.y);

Test.x = 99;

Console.WriteLine(Test.x);

}

}

/\*

Output:

0

5

99

\*/

# Classi statiche e membri di classi statiche

http://msdn.microsoft.com/it-it/library/79b3xss3.aspx

Una classe statica corrisponde fondamentalmente a una classe non statica, ma c'è una differenza: di una classe statica non è possibile creare un'istanza. In altre parole, non è possibile utilizzare la parola chiave new per creare una variabile del tipo di classe. Poiché non esiste una variabile dell'istanza, si accede ai membri di una classe statica tramite il nome stesso della classe. Ad esempio, se si dispone di una classe statica denominata UtilityClass che dispone di un metodo pubblico denominato MethodA, si chiama il metodo come mostrato nell'esempio seguente:

UtilityClass.MethodA();

Una classe statica può essere utilizzata come un contenitore adatto per insiemi di metodi che funzionano solo sui parametri di input e non devono ottenere o impostare campi di istanza interni. Ad esempio, nella libreria di classi .NET Framework, la classe statica System.Math contiene diversi metodi che eseguono operazioni matematiche, senza alcun requisito per archiviare o recuperare dati univoci per una particolare istanza della classe Math. In altre parole, si applicano i membri della classe specificando il nome della classe e il nome del metodo, come illustrato nell'esempio seguente.

double dub = -3.14;

Console.WriteLine(Math.Abs(dub));

Console.WriteLine(Math.Floor(dub));

Console.WriteLine(Math.Round(Math.Abs(dub)));

// Output:

// 3.14

// -4

// 3

Come per tutti i tipi di classi, le informazioni sul tipo di una classe statica vengono caricate da .NET Framework Common Language Runtime (CLR) durante il caricamento del programma che fa riferimento alle classi. Il programma non può specificare esattamente quando la classe verrà caricata. Tuttavia, è garantito che la classe sarà caricata, che i suoi campi saranno inizializzati e che il costruttore statico sarà chiamato prima che il programma faccia riferimento per la prima volta alla classe stessa. Un costruttore statico viene chiamato solo un volta e una classe statica rimane in memoria per la durata del dominio dell'applicazione in cui risiede il programma.

Nel seguente esempio sono riportate le principali funzionalità delle classi statiche:

* Contiene solo membri statici.
* Non è possibile crearne istanze.
* È sealed.
* Non può contenere Costruttori di istanze.

La creazione di una classe statica è pertanto simile alla creazione di una classe che contiene solo membri statici e un costruttore privato, che impedisce la creazione di istanze della classe. Una classe statica presenta un indubbio vantaggio. Consente infatti al compilatore di verificare che non vengano aggiunti accidentalmente membri di istanze e quindi di garantire che non vengano create istanze di questa classe.

Le classi statiche sono sealed e pertanto non possono essere ereditate. Possono ereditare solo dalla classe Object. Le classi statiche non possono contenere un costruttore di istanza; possono però contenere un costruttore statico. Classi non statiche devono definire anche un costruttore statico se la classe contiene membri statici che richiedono un'inizializzazione più complessa. Per ulteriori informazioni, vedere Costruttori statici (Guida per programmatori C#).

**Esempio:**

Di seguito è riportato un esempio di una classe statica contenente due metodi che consentono di convertire i valori relativi alla temperatura da gradi Celsius a gradi Fahrenheit e viceversa:

public static class TemperatureConverter

{

public static double CelsiusToFahrenheit(string temperatureCelsius)

{

// Convert argument to double for calculations.

double celsius = Double.Parse(temperatureCelsius);

// Convert Celsius to Fahrenheit.

double fahrenheit = (celsius \* 9 / 5) + 32;

return fahrenheit;

}

public static double FahrenheitToCelsius(string temperatureFahrenheit)

{

// Convert argument to double for calculations.

double fahrenheit = Double.Parse(temperatureFahrenheit);

// Convert Fahrenheit to Celsius.

double celsius = (fahrenheit - 32) \* 5 / 9;

return celsius;

}

}

class TestTemperatureConverter

{

static void Main()

{

Console.WriteLine("Please select the convertor direction");

Console.WriteLine("1. From Celsius to Fahrenheit.");

Console.WriteLine("2. From Fahrenheit to Celsius.");

Console.Write(":");

string selection = Console.ReadLine();

double F, C = 0;

switch (selection)

{

case "1":

Console.Write("Please enter the Celsius temperature: ");

F = TemperatureConverter.CelsiusToFahrenheit(Console.ReadLine());

Console.WriteLine("Temperature in Fahrenheit: {0:F2}", F);

break;

case "2":

Console.Write("Please enter the Fahrenheit temperature: ");

C = TemperatureConverter.FahrenheitToCelsius(Console.ReadLine());

Console.WriteLine("Temperature in Celsius: {0:F2}", C);

break;

default:

Console.WriteLine("Please select a convertor.");

break;

}

// Keep the console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

/\* Example Output:

Please select the convertor direction

1. From Celsius to Fahrenheit.

2. From Fahrenheit to Celsius.

:2

Please enter the Fahrenheit temperature: 20

Temperature in Celsius: -6.67

Press any key to exit.

\*/

**Membri statici**

Una classe non statica può contenere metodi, campi, proprietà o eventi statici. È possibile chiamare il membro statico di una classe anche quando non sono state create istanze della classe. Al membro statico si accede sempre tramite il nome della classe, non tramite il nome dell'istanza. Di un membro statico esiste una sola copia, indipendentemente dal numero di istanze della classe create. Proprietà e metodi statici non possono accedere a campi non statici ed eventi nel tipo che li contiene e non possono accedere a una variabile dell'istanza di qualsiasi oggetto a meno che non venga esplicitamente passata in un parametro del metodo.

È più frequente dichiarare una classe non statica con alcuni membri statici, che dichiarare un'intera classe come statica. Due utilizzi comuni di campi statici sono: tenere un conteggio del numero di oggetti di cui è stata creata un'istanza o archiviare un valore che deve essere condiviso fra tutte le istanze.

I metodi statici possono essere sottoposti a overload ma non a override, perché appartengono alla classe e non a qualsiasi istanza della classe.

Sebbene non sia possibile dichiarare un campo come static const, il campo const è essenzialmente statico nel comportamento. Appartiene al tipo, non a istanze del tipo. Pertanto, è possibile accedere ai campi const tramite la stessa notazione ClassName.MemberName utilizzata per i campi statici. Non è richiesta alcuna istanza dell'oggetto.

C# non supporta variabili locali statiche (variabili dichiarate nell'ambito del metodo).

I membri di classe statici vengono dichiarati utilizzando la parola chiave static prima del tipo restituito, come illustrato nel seguente esempio:

public class Automobile

{

public static int NumberOfWheels = 4;

public static int SizeOfGasTank

{

get

{

return 15;

}

}

public static void Drive() { }

public static event EventType RunOutOfGas;

// Other non-static fields and properties...

}

I membri statici vengono inizializzati prima dell'accesso iniziale e prima dell'eventuale chiamata al costruttore statico, se presente. Per accedere a un membro di una classe statica, utilizzare il nome della classe anziché il nome di una variabile per specificare la posizione del membro, come illustrato nel seguente esempio:

Automobile.Drive();

int i = Automobile.NumberOfWheels;

Se la classe contiene campi statici, fornire un costruttore statico che li inizializzi al caricamento della classe.