Academy .NET

Tipo Struct e Record

Sommario

- Le struct
- I record
- Tipo dynamic

Le struct

Il tipo **struct** è un **tipo valore** che deriva da **ValueType**, **ValueType** deriva a sua volta da **System.Object.**

Tutti i tipi primitivi in **.NET Framework** sono definiti come Struct.

Le struct

La definizione di una Struct è simile alla definizione di una classe, con diverse differenze.

Le struct **non** supportano l'ereditarietà.

Si utilizzano principalmente per la creazione di una struttura semplice che occupa poca memoria.

- Es: un tipo numerico personalizzato

Definizione di una struct

In genere si definisce una **struct** quando si necessita una struttura dati che rappresenta un insieme di **tipi** semplici da raggruppare e organizzare in un solo tipo.

```
struct <NomeStruct>
{
//membri struct
}
```

modificatoreAccesso tipo nome;

Le struct

- Si utilizza il modificatore **public** per ogni campo

```
struct Point3D
{
public double a;
public double b;
public double c;
}
```

- Dichiarazione variabile di tipo struct

```
Point3D punto;
```

Le struct

Le struct, come detto in precedenza, sono di **tipo valore.**

É possibile richiamare il costruttore di default con **new** ma non è richiesto:

```
Point3D p;
p.a = 1;
p.b = 2;
p.c = 3;
```

Le struct

Nel caso di costruzione dell'oggetto tramite invocazione del costruttore, il valore dei campi sarebbe il corrispettivo valore di **default** del proprio tipo**.**

```
Point3D point = new Point3D() // a = 0; b = 0; c = 0;
```

Le struct

Non è possibile assegnare un valore a un campo di tipo struct.

```
struct Point3D
{
public double a=1; //errore
public double b;
public double c;
}
```

Le struct

Le struct, come le classi, supportano costruttori personalizzati con parametri.

Non è possibile definire in modo esplicito il **costruttore di default**

Le struct

É possibile, ma non necessario, utilizzare la parola chiave **this.**

L'inizializzazione di tutti i campi è obbligatoria

```
public Point(double a, double b, double c)
{
this.a = a;
this.b = b;
this.c = c;

}
```

Le struct

Assegnare una variabile di tipo struct a un'altra variabile c**rea una copia del valore**

Con le **classi** si copierebbe esclusivamente il riferimento all'oggetto.

Struct

La **struct** può contenere:

- Campi
- Costanti
- Eventi
- Proprietà
- Metodi
- Costruttori
- Indicizzatori
- Operatori
- Tipi innestati

Struct

La **struct** non può contenere:

- Distruttori
- Costruttori senza parametri

I record

I record sono una nuova funzionalità di C# 9.0

Supportati da **.NET 5**

- I **record** sono di tipo **riferimento**
- Simili a classi
- Campi di **sola lettura**

I record

- I record non sono modificabili

I record

```
public record Record
{
    public string firstField { get; }
    public string secondField { get; }

    public Record(string first,
                  string last) => (firstField, secondField) = (first, last);
}
```

- Record è un tipo riferimento
- Le due proprietà firstField e secondField **non** sono modificabili.
- I record supportano l'ereditarietà

I record

Metodi presenti per ogni tipo record:

- Metodi per confronti **value-based**
- Override per **GetHashCode()**
- Metodi per copia e clonazione
- **ToString()**

I record

- Se il tipo di record è **abstract**, anche il metodo "clone" è astratto. Se il tipo di base non è object, il metodo sarà **override**.

- Per i tipi di record che non sono abstract quando il tipo di base è object:
  - Se il record è **sealed** (non ereditabile), nessun modificatore aggiuntivo viene aggiunto al metodo "**clone**".
  - Se il record non è sealed, il metodo "clone" è virtuale.

- Per i tipi di record che non sono **abstract** quando il tipo base non è **object:**
  - Se il record è sealed, sarà sealed anche il metodo "clone".
  - Se il record non è sealed, il metodo "clone" viene sovrascritto.

I record

Due record sono uguali fra loro se le loro proprietà e il loro tipo sono uguali.

Tipo dynamic

La parola chiave **dynamic** è stata introdotta in C# 4.0.

Indica un nuovo tipo : il tipo **dynamic**.

La variabile **dynamic** può mantenere il riferimento a un oggetto di qualunque tipo supportato da C#. Possono essere di tipo dynamic variabili, parametri, proprietà, tipi di ritorno ecc.

Tipo dynamic

Il tipo dynamic internamente viene convertito implicitamente da e verso qualunque altro tipo .NET.

Il tipo dynamic è simile al tipo Object con la differenza che il compilatore evita di effettuare il controllo sui tipi, che vengono ricavati e riconosciuti a **tempo di esecuzione** . Successivamente il CLR si occuperà di utilizzare operazioni tra tipi, come per esempio la somma nel seguente esempio:

```
dynamic a = 1;
dynamic b = 3;
dynamic c = a + b;

Console.WriteLine("La somma è: " + c);
```

Tipo dynamic

Non è disponibile Intellisense per il tipo dynamic in quanto non è possibile riconoscerne il tipo.

Se si tentasse di eseguire una operazione specrifica per tipi non supportati (per esempio divisione tra stringhe) il compilatore non effettuerà nessun controllo come ci si aspetterebbe. In ogni caso verrò lanciata una eccezione a tempo di esecuzione di tipo **RuntimeBinderException**.

Il tipo dynamic riinvia la verifica dei tipi al momento dell'esecuzione.

Tipo dynamic

Importante è non confondere il tipo **dynamic** con il tipo **var**.

var indica che una variabile assumerà implicitamente un tipo, che però il compilatore conosce. Il tipo **dynamic** invece sarà determinato a tempo di esecuzione.

Domande & approfondimenti

Academy .NET