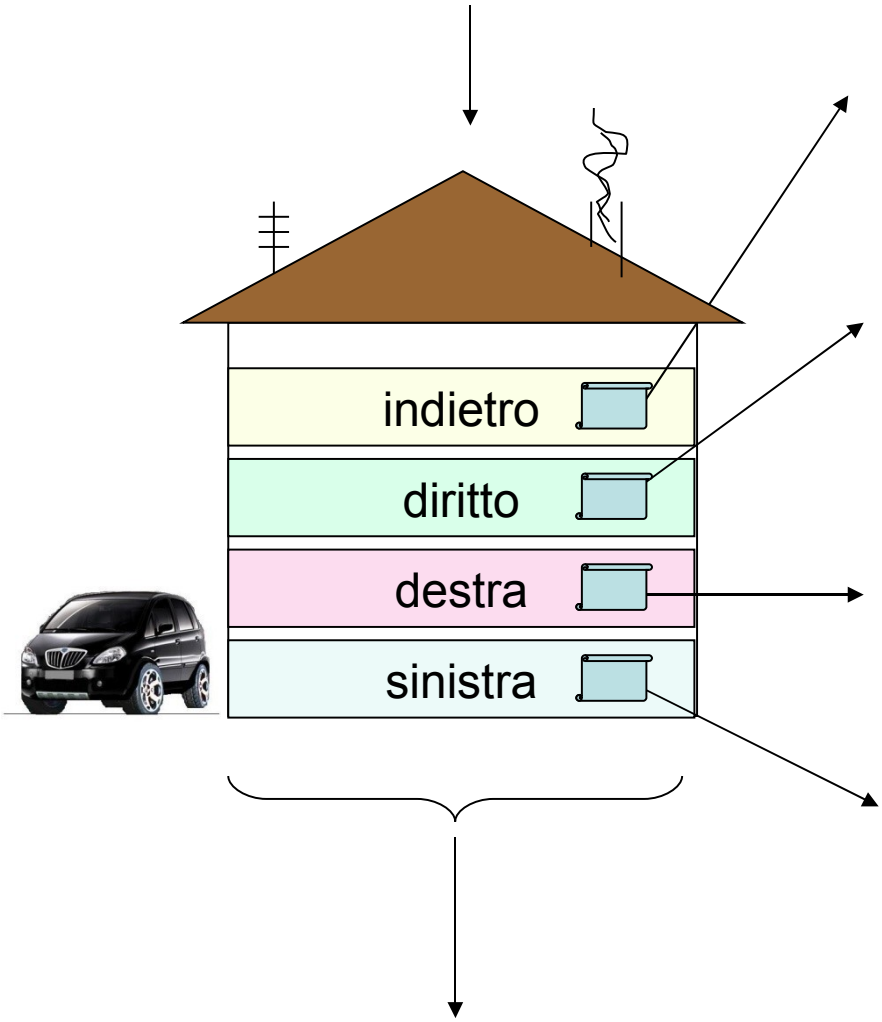
Io sono il tipo enumerativo TipoDirezione
e rappresento l'intera casa



Io sono al terzo piano e valgo 3 ma se mi chiami con il metodo toString() ti restituisco il mio nome ossia indietro.

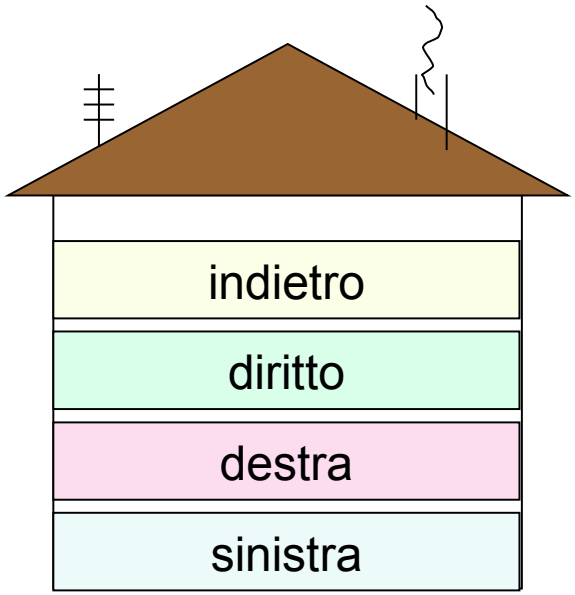
Io sono al secondo piano e valgo 2 ma se mi chiami con il metodo toString() ti restituisco il mio nome ossia diritto.

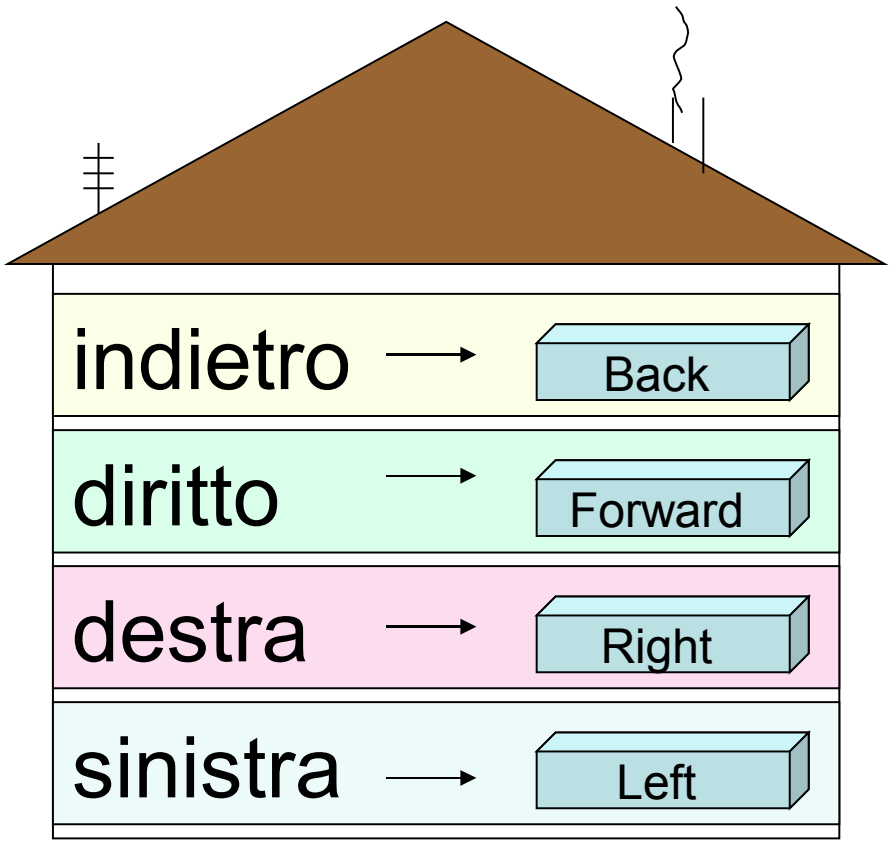
Io sono al primo piano e valgo 1 ma se mi chiami con il metodo toString() ti restituisco il mio nome ossia destra.

Io sono al piano terra e valgo 0 ma se mi chiami con il metodo toString() ti restituisco il mio nome ossia sinistra.

Con il metodo GetType() ti dico che sono di tipo enumerativo TipoDirezione

```
namespace fbblTipi
{
    public class StrumentiTipi
    {
        public enum TipoDirezione
        {
            sinistra,
            destra,
            diritto,
            indietro
        };
    }
}
```





C’è una stanza chiamata “Back”

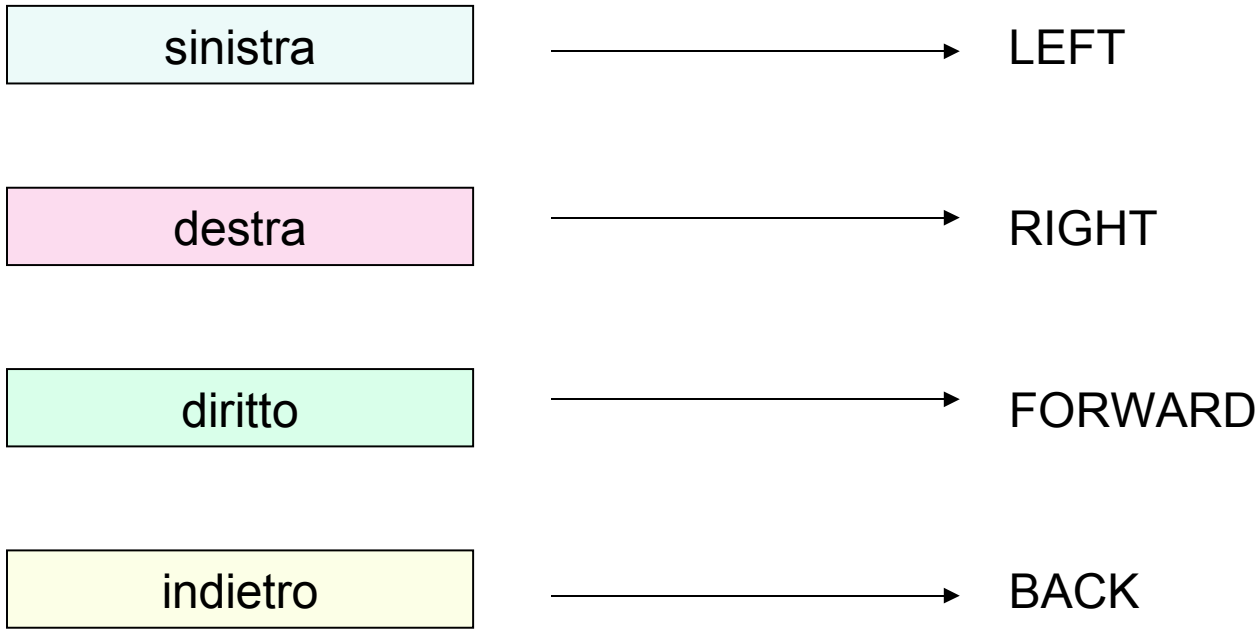
C’è una stanza chiamata “Forward”

C’è una stanza chiamata “Right”

C’è una stanza chiamata “Left”

Come è possibile creare “delle stanze” nei vari “piani” del tipo enumerativo?

IN SOSTANZA CI CHIEDIAMO COME POSSIAMO ASSOCIARE UNA STRINGA PERSONALIZZATA AD OGNI CAMPO DEL TIPO ENUMARATIVO ?



La riposta la trovate a partire dal .NET 3.5

In particolare nelle classi:

System.Reflection
System.ComponentModel

```
using System.ComponentModel;
using System.Reflection;

namespace fbblTipi
{
    public class StrumentiTipi
    {
        public enum TipoDirezione
        {
            [Description("LEFT")]
            sinistra,
            [Description("RIGHT")]
            destra,
            [Description(“FORWARD")]
            diritto,
            [Description("BACK")]
            indietro
        };
    }
}
```

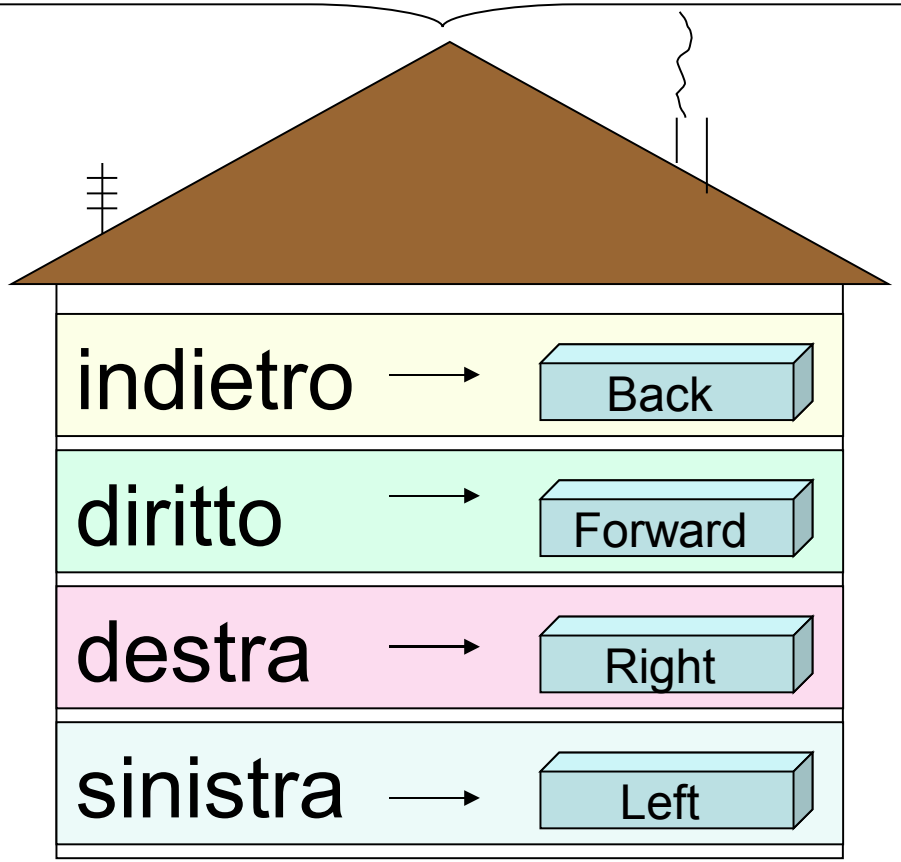
Le classi che mi permettono di inserire le stanze nei vari piani del tipo enumerativo

Definisco le varie stanze nei rispettivi piani

Devo usare [Description(“nome-stanza”)] e prima del nome che identifica il piano

Come faccio ad estrapolare le stanze dei vari piani presenti nel tipo enumerativo chiamato TipoDirezione?

Non dimenticate mai che l'enumerativo TipoDirezione è una struttura dati complessa indivisibile paragonabile ad una casa



Per estrapolare la stanza di un piano dell'enumerativo TipoDirezione devo seguire i seguenti passi:

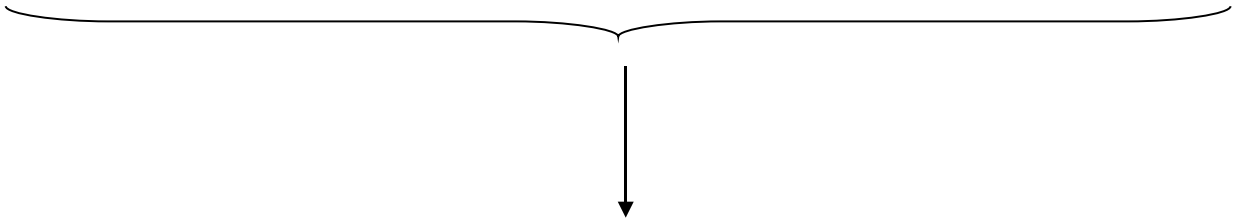
1. Capire in che modo è stato costruito il piano che stiamo analizzando chiedendo a "qualcuno" di dirci cosa c'è al piano che vogliamo verificare.

3. Questo qualcuno è il metodo "GetField("nome-piano")" che lavora sul tipo enumerativo restituito dal metodo GetType().

5. Tutte le informazioni del piano che vengono restituite dal metodo "GetField()" devono venire memorizzate in un contenitore particolare di tipo classe "FieldInfo".

Segue il codice esplicativo.....

```
Enum MiaDirezione;
MiaDirezione = StrumentiTipi.TipoDirezione.diritto;
```



Il primo step è dichiarare una variabile di tipo classe Enum che chiamo “MiaDirezione”. Poi assegno a questa variabile il tipo enumerativo “TipoDirezione” indicando uno dei quattro campi, che in questo esempio è “diritto”.
Non dimenticate che il tipo enumerativo “TipoDirezione” è stato inserito all’ interno del namespace “StrumentiTipi”.

```
FieldInfo InformazioniSulPiano;
```

Il secondo o step è dichiarare una variabile di tipo classe “FieldInfo” che permetterà di memorizzare tutte le informazioni di un piano della struttura enumerativa “TipoDirezione”. Per comodità ho chiamato questa variabile “InformazioniSulPiano”.

```
InformazioniSulPiano = MiaDirezione.GetType().GetField("destra");
```

1. Devo selezionare l’intera casa ossia il tipo della struttura enumerativa da cui voglio estrapolare le informazioni. Per restituire il tipo della variabile “MiaDirezione” devo usare il metodo “GetType()”. Questo passaggio può essere visto come:

   “Usa la GetType() per leggere il progetto dell’intera casa……”

InformazioniSulPiano = MiaDirezione.GetType().GetField("destra");

1. Devo selezionare l'intera casa ossia il tipo della struttura enumerativa da cui voglio estrapolare le informazioni. Per restituire il tipo della variabile "MiaDirezione" devo usare il metodo "GetType()". Questo passaggio può essere visto come:

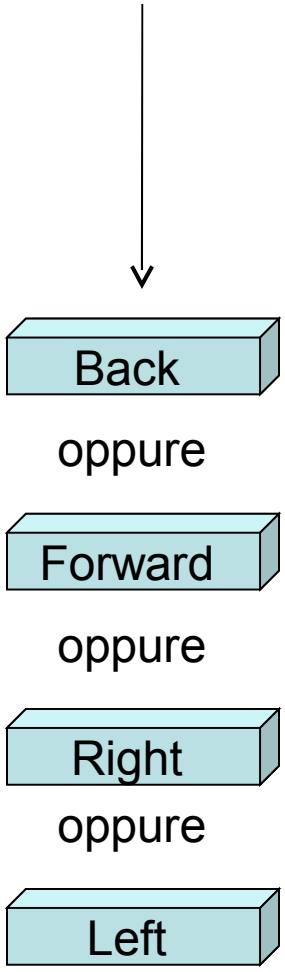
"Uso la GetType() per leggere il progetto dell'intera casa……"

1. Nel momento in cui conosco come è realizzata la struttura enumerativa "MiaDirezione" ossia conosco il progetto della casa, posso focalizzare la mia attenzione su un determinato piano utilizzando il metodo "GetField("nome-piano")" il quale restituisce una serie di informazioni particolari che non possono venire memorizzate in semplici variabili di tipo stringa ma devono venire memorizzate in una variabile complessa di tipo classe "FieldInfo" e quindi nella variabile dichiarata precedentemente chiamata "InformazioniSulPiano".

Come posso ora leggere tutte le informazioni memorizzate nella variabile complessa “InformazioniSulPiano” di tipo classe “FieldInfo”?

Prima di rispondere a questa domanda dobbiamo avere chiaro che le INFORMAZIONI memorizzate nella variabile di tipo classe “FieldInfo” sono attributi particolari scelti e definiti dal programmatore, nel gergo informatico si chiamano CUSTOM ATTRIBUTES!

Quali sono i CUSTOM ATTRIBUTES che la variabile “InformazioniSulPiano” di tipo classe “FieldInfo” ha memorizzato?



La risposta è una delle quattro stanze in base al piano analizzato….

Per estrapolare i CUSTOM ATTRIBUTES da un generico oggetto si utilizza la funzione chiamata:

GetCustomAttributes()

Nel gergo informatico usiamo il termine QUERY OUR CUSTOM ATTRIBUTES per l'estrapolazione dei CUSTOM ATTRIBUTES da un generico oggetto:

TYPE_OF_OBJECT.GetCustomAttributes()

Nel nostro esempio l'oggetto è rappresentato dalla variabile di tipo classe "FieldInfo" chiamata "InformazioniSulPiano" ma potremmo avere altri tipo di oggetti che hanno memorizzato al loro interno dei CUSTOM ATTRIBUTES che vogliamo estrapolare ecco quindi che in tutti i casi useremo la funzione "GetCustomAttributes()" variando il tipo di oggetto su cui lavoriamo.

In poche parole la funzione GetCustomAttributes() è di uso generico per molti oggetti del Framework .NET per l'estrapolazione dei corrispondenti CUSTOM ATTRIBUTES

Abbiamo detto che “GetCustomAttributes()” è una funzione quindi procediamo a vedere cosa restituisce e come è fatto il prototipo della suddetta.

Scopriremo che la “GetCustomAttributes()” è POLIMORFICA come moltissime funzioni del Framework .NET

```
public object[] GetCustomAttributes
                (
                   Type attributeType,
                   bool inherit
                );

public object[] GetCustomAttributes
                (
                  bool inherit
                );
```

```
public object[] GetCustomAttributes
                (
                  Type attributeType,
                  bool inherit
                );
```

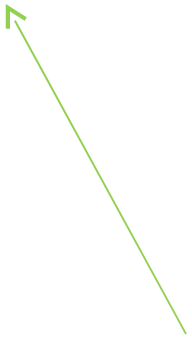
1. La funzione restituisce un vettore o array di tipo “object” perché essendo questa di uso generico per molti oggetti del Framework .NET non è possibile sapere a priori quali tipi di CUSTOM ATTRIBUTES la funzione va a leggere quindi per non incorrere in errori si identifica il dato restituito di tipo “object” che è la classe comune a tutto il Framework .NET ossia il padre di tutta la gerarchia di oggetti.
Sarà poi compito del programmatore effettuare il CASTING del tipo restituito.
Viene restituito un vettore perché non sappiamo a priori il numero di CUSTOM ATTRIBUTES che la funzione leggerà quindi dovremo successivamente verificare il numero di attributi letti.

1. Il primo parametro in ingresso “Type attributeType” informa la funzione sul tipo di CUSTOM ATTRIBUTES che deve leggere.
Diviene fondamentale dire alla funzione il tipo di informazione che legge proprio per il fatto che diversi oggetti memorizzeranno informazioni diverse.
Nel nostro caso dovremo dire il tipo rappresentato dalla “stanza presente sul piano”.

1. Il secondo parametro in ingresso “bool inherit” essendo booleano può valere solo “true” oppure “false” e serve per permettere alla funzione di ricercare in tutta la struttura gerarchica il tipo di dati letto nel caso non riuscisse a riconoscerlo.
La ricerca nella struttura gerarchica avviene solo se questo parametro in ingresso vale “true”.
Nel nostro caso imposteremo “false” perché definiremo noi in modo chiaro il tipo di CUSTOM ATTRIBUTES che la funzione andrà a leggere quindi non diviene necessaria nessuna ricerca per l’identificazione del tipo di dato.
Ricordo che il tipo di attributo letto corrisponde alla “stanza presente sul piano”.

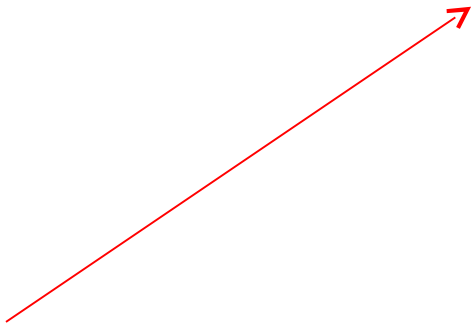
Procediamo alla creazione in C# dell'istruzione per l'estrapolazione dei CUSTOM ATTRIBUTES in base a tutto quello che abbiamo visto fino a questo momento.

… InformazioniSulPiano.GetCustomAttributes(… , …)



1. Usiamo la funzione “GetCustomAttributes” sull’oggetto “InformazioniSulPiano” che sappiamo essere un’istanza della classe “FieldInfo” da cui vogliamo estrarre l’unico CUSTOM ATTRIBUTES presente corrispondente alla “stanza sul piano”.

… InformazioniSulPiano.GetCustomAttributes(typeof(DescriptionAttributes) , …)

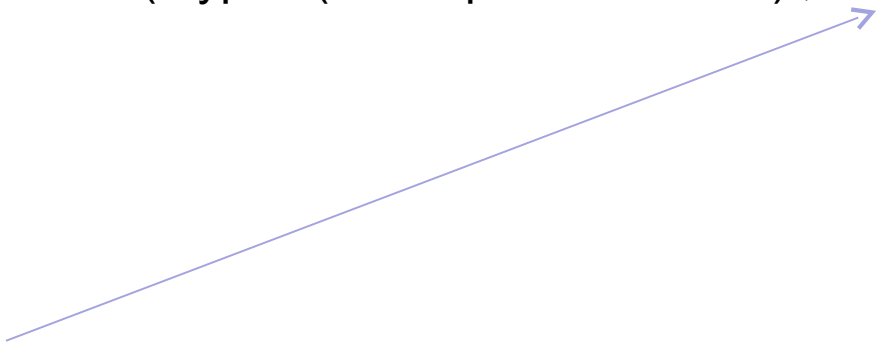


1. Grazie al primo parametro in ingresso diciamo alla funzione il tipo di CUSTOM ATTRIBUTES che dovrà leggere dall'oggetto "InformazioniSulPiano".
Le stanze che abbiamo definito nel tipo enumerativo "TipoDirezione" sono CUSTOM ATTRIBUTES di tipo "DescriptionAttributes".
L'operatore "typeof" ha il compito di restituire tutte le informazioni sul tipo "DescriptionAttributes" in modo che la funzione possa avere tutte le credenziali per leggere correttamente il CUSTOM ATTRIBUTES presente.

Perché la “stanza del piano” è di tipo “DescriptionAttributes”?

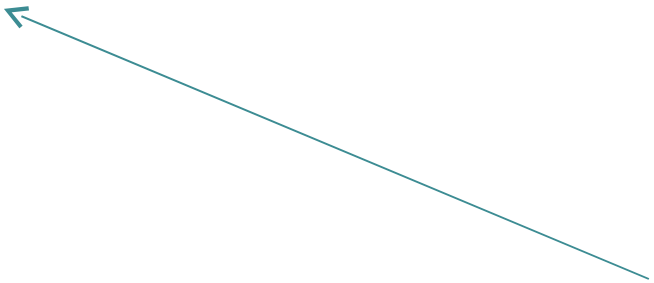
Perché tutto ciò che è definito nell’attributo [Description(“custom-attribute”)] è di tipo “DescriptionAttributes”.

… InformazioniSulPiano.GetCustomAttributes(typeof(DescriptionAttributes) , false)



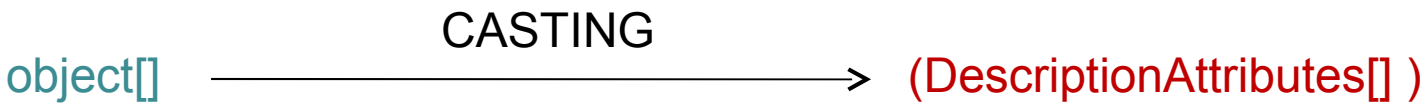
3. Il secondo parametro in ingresso di tipo booleano impostato a “false” fa si che la funzione non cerchi nella struttura gerarchica il tipo di dato del CUSTOM ATTRIBUTES perché abbiamo già definito noi il tipo nel primo parametro in ingresso.

object[] InformazioniSulPiano.GetCustomAttributes(typeof(DescriptionAttributes),false)



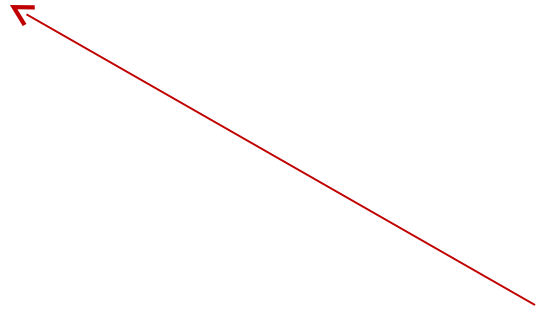
4. La funzione come sappiamo restituisce un vettore e dal prototipo sappiamo che ogni elemento dell'array è in generale identificato dal tipo generico "object" come ho riportato nell'istruzione sopra stante.

Il prototipo usa "object" proprio per accumunare tutti i casi possibili, quindi nel nostro caso dovremo effettuare la conversione chiamata CASTING da vettore di tipo "object" a vettore di tipo "DescriptionAttributes" perché il tipo di attributo letto è proprio questo. Non dimenticate che il CASTING prevede l'uso delle parentesi tonde.



Il lettore attento potrebbe chiedersi perché usare un vettore se nel piano ho una sola stanza ossia un solo CUSTOM ATTRIBUTES. Il prototipo della funzione prevede la restituzione di un vettore e quindi non possiamo astenerci da questo vincolo.
Il massimo che possiamo fare è quello del CASTING ossia adattare il tipo "object" al tipo del nostro CUSTOM ATTRIBUTES ma nulla di più.

(DescriptionAttributes[]) InformazioniSulPiano.GetCustomAttributes(typeof(DescriptionAttributes),false)



5. La funzione deve restituire quindi un vettore di tipo “DescriptionAttributes” grazie al CASTING.

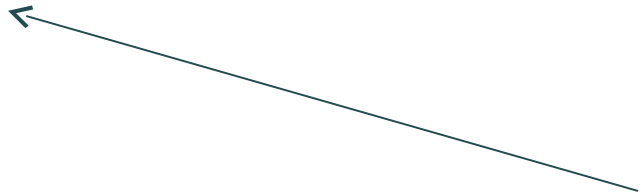
Non dimentichiamo che siamo in presenza di una funzione il tipo di dato restituito deve finire nel LEFT VALUE.

LEFT_VALUE = (DescriptionAttributes[]) InformazioniSulPiano.GetCustomAttributes(typeof(DescriptionAttributes),false)

```
DescriptionAttribute[]  miastanza;

…

miastanza = (DescriptionAttributes[])  InformazioniSulPiano.GetCustomAttributes(typeof(DescriptionAttributes),false);
```



6. Definisco un vettore di tipo “DescriptionAttributes” che sappiano avrà un solo elemento riferito alla stanza del piano che abbiamo analizzato rappresentato dalla variabile “miastanza”.

Ora che abbiamo estrapolato il nostro CUSTOM ATTRIBUTES e memorizzato nella variabile vettore “miastanza”,potremmo procedere subito a stampare il suo valore, ma conviene prima controllare che il vettore non sia vuoto e procedure successivamente alla visualizzazione del CUSTOM ATTRIBUTES.

Scriviamo la funzione personalizzata chiamata ElencoAttributi con il relativo metodo di chiamata.

```
public void sinistra()
    {
        string direzione;

        direzione = StrumentiSimulazione.ElencoAttributi(StrumentiTipi.TipoDirezione.sinistra);

        this.set_Direzione(direzione);

        Console.WriteLine("L'autovettura " + this.get_ModelloVeicolo() + …… );
    }
```

passo la variabile enumerativa
con il valore “sinistra”

Nel metodo sinistra andiamo a richiamare la funzione “ElencoAttributi” la quale ha come unico parametro in ingresso uno dei quattro valori del tipo enumerativo.

```
public static string ElencoAttributi(Enum PianoDaAnalizzare)
    {
        FieldInfo InformazioniSulPiano= PianoDaAnalizzare.GetType().GetField(PianoDaAnalizzare.ToString());

        DescriptionAttribute[] miastanza = (DescriptionAttribute[]) InformazioniSulPiano.GetCustomAttributes(typeof(DescriptionAttribute), false);

        return (miastanza.Length > 0) ? miastanza[0].Description : "Errore generico nella lettura dei custom attributes";
    }
```

mi restituisce il nome del piano da analizzare come se ci fosse scritto GetField("sinistra")

Utilizzo un'espressione nella return infatti "miastanza.Length>0" è l'equivalente di una condizione nel controllo condizionale "if" e se questa è vera viene eseguita la prima istruzione a destra del "?" ossia "miastanza[0].Description" altrimenti viene eseguita l'istruzione a destra dei ":" ossia il messaggio "Errore generico….".

Inutile dire che miastanza è un vettore e che essendoci un solo elemento questo si troverà nella prima posizione dell'array identificata dall'indice pari a zero.
Il vettore contiene una classe di tipo "DescritptionAttributes" ossia un dato complesso il quale ha un CUSTOM ATTRIBUTES identificato proprio dal campo "Description".