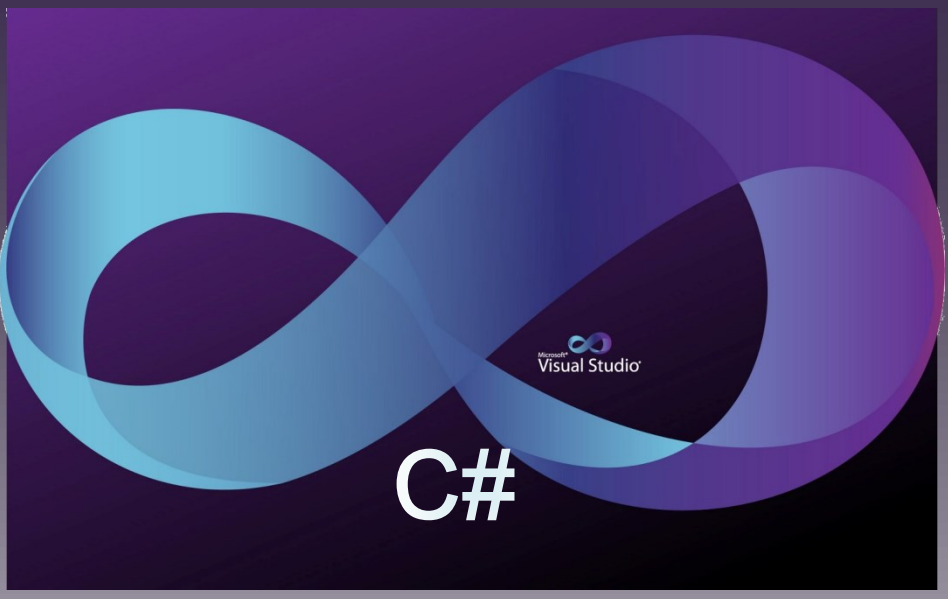
**ISTITUTO TECNICO INDUSTRIALE**
**G. M. ANGIOY**
**SASSARI**




# CORSO DI PROGRAMMAZIONE

## USO DELLE PROPRIETÀ

## DISPENSA 01.02

01-02_Proprietà_[ver_15]



Versione del: 22/09/2022
Revisione numero: 15

Prof. Andrea Zoccheddu
Dipartimento di Informatica

**DIPARTIMENTO**
**INFORMATICA E TELECOMUNICAZIONI**

# PROPRIETÀ DEI CONTROLLI

## COS'È UNA PROPRIETÀ?

### CONTROLLI E PROPRIETÀ

Le proprietà sono dei contenitori di dati simili alle variabili, che offrono un meccanismo flessibile per la lettura, la scrittura o il calcolo dei valori delle caratteristiche di oggetti. Possono essere utilizzate come se fossero variabili, ma sono in realtà dei gestori di valori che sfruttano delle funzioni di accesso. In questo modo, è possibile usare facilmente i dati pur conservando sicurezza e flessibilità.

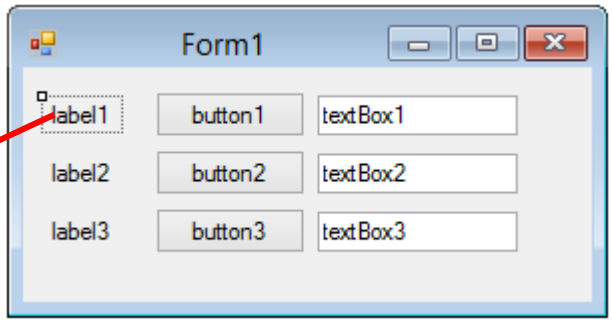
Ogni controllo ha delle proprietà; alcune sono comuni a tutti i controlli, altre sono specifiche di un determinato controllo o gruppo di controlli.

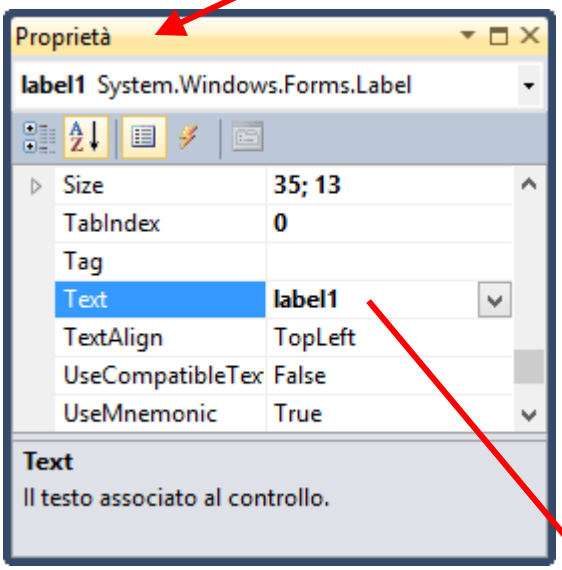
LINK

Fonte: Microsoft http://msdn.microsoft.com/it-it/library/67ef8sbd(v=vs.90)

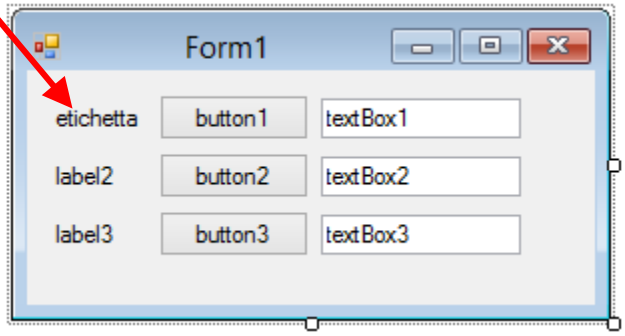
**PROGETTO GUIDATO**

- Crea un progetto visuale e predisponi i controlli come nella figura seguente:
- Nel Form ci sono 3 **`label`**, 3 **`textBox`**
- Seleziona la **`label1`**



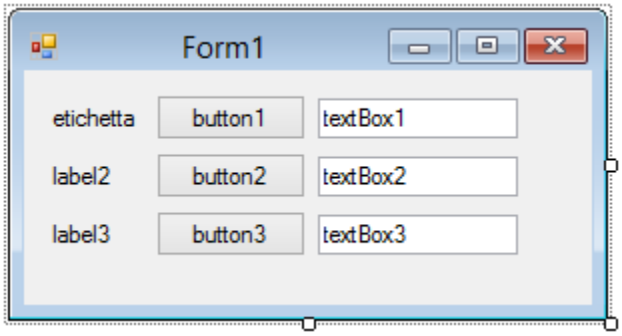

- Quando si seleziona un controllo (es. **`label1`**) la finestra delle proprietà (di solito in basso a destra) mostra le proprietà del controllo scelto (nel nostro caso di **`label1`**).
- Nella finestra delle propietà del controllo una proprietà è scelta come caratteristica attiva e pronta ad essere modificata (di solito è l'ultima modificata oppure quella di base come **`Text`**).
- Se si desidera modificare un'altra proprietà è sufficiente selezionarla col mouse.
- Se la proprietà Text è selezionata e si digita la frase **`etichetta`** e infine premendo invio, si ottiene che l'aspetto del controllo cambia e compare la scritta sul controllo.
- In generale la modifica di una proprietà modifica l'aspetto o il comportamento del controllo.



## CONCETTO DI CONTROLLO

Abbiamo visto che in un Form è possibile inserire molti controlli diversi. Sebbene alcuni controlli siano differenti tra loro, tuttavia è possibile inserirne molti ma dello stesso tipo. Per esempio è possibile inserire molti **button** oppure molte **label** oppure molte **textBox**.

- Nella finestra illustrata qui di lato (relativa all'esempio precedente) per esempio sono presenti 3 button rispettivamente denominati `button1`, `button2` e `button3`.
- I tre button sono diversi tra loro (hanno posizione, scritte e reazioni diverse) ma sono tutti dello stesso tipo: `button`.

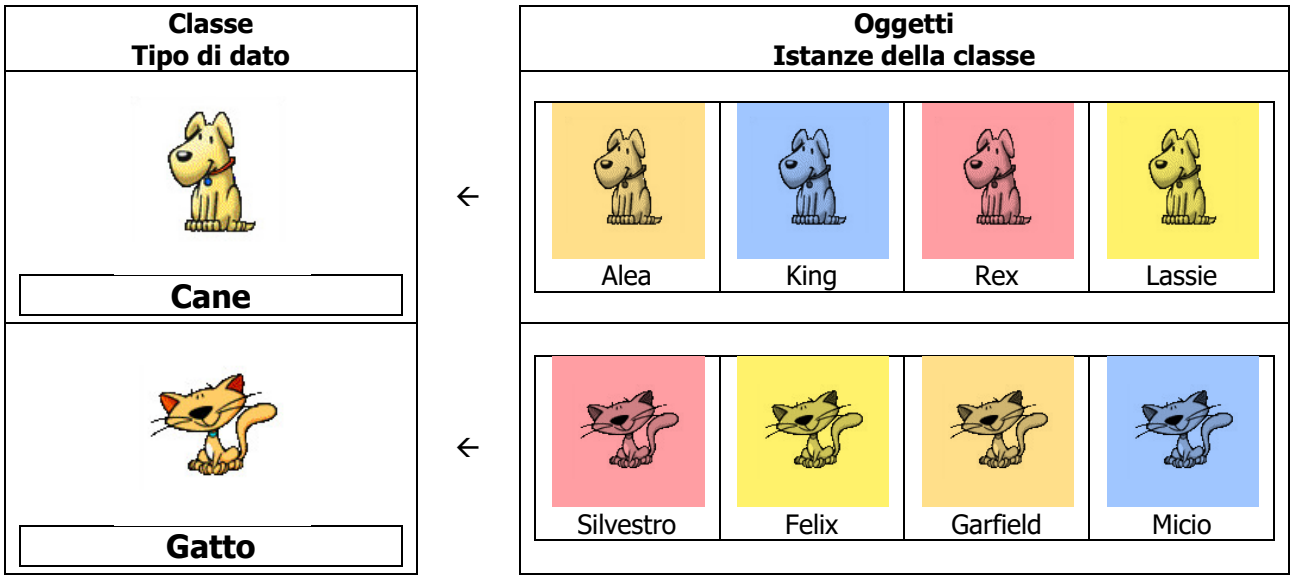



Analogamente le tre label sono tutte di tipo Label e le tre textBox sono tutte di tipo TextBox:

| Classe | Controllo |
|---|---|
| `Button` | button1, button2, button3, . . . |
| `Label` | label1, label2, label3, . . . |
| `TextBox` | textBox1, textBox2, textBox3, . . . |

In un ambiente come Visual Studio i controlli sono casi particolari di elementi detti oggetti (sono casi particolari di oggetto) e i tipi del controllo sono dette classi. In generale ogni controllo deve appartenere ad una classe. La classe determina l'aspetto, il comportamento, la funzione e il modo di gestire un oggetto. Ogni singolo oggetto invece è un caso particolare di quell'oggetto.

Ad esempio potremmo pensare alle classi Cane e Gatto a cui appartengono i diversi cani e gatti del nostro programma.

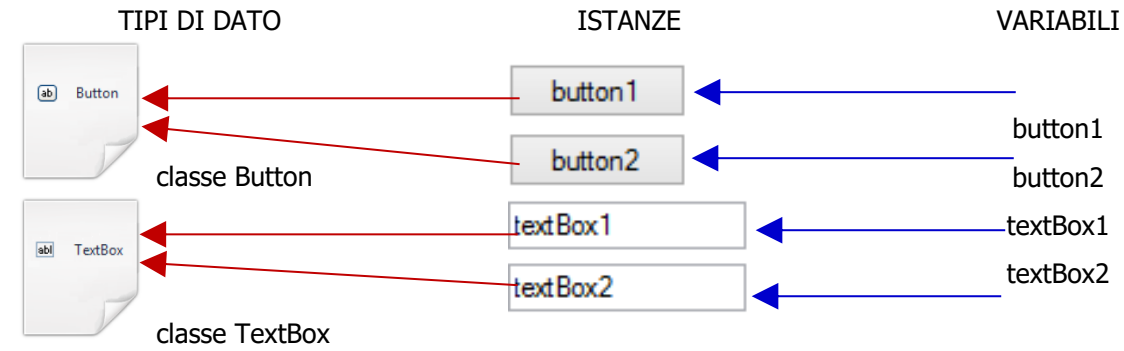


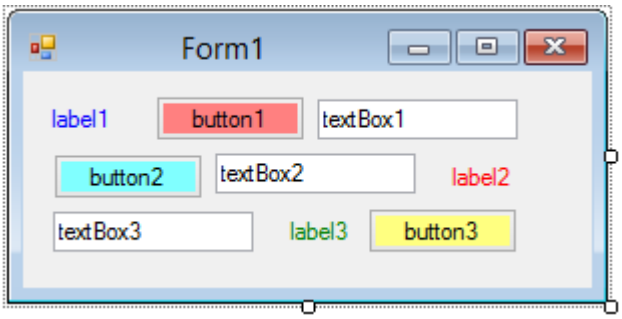

Come nello schema sopra illustrato si ipotizza di avere la classe Cane a cui appartengono 4 cani diversi: Alea, King, Rex, Lassie. Ciascuno di questi cani è un cane; di conseguenza ogni cane ha caratteristiche analoghe a quelle degli altri cani, tra cui informazioni (il nome, la razza, il colore, il sesso, ecc.) e comportamenti (abbaiano, latrano, fanno la guardia, uggiolano). Analogamente è proposta la classe Gatti a cui appartengono tanti gatti.

La classe o tipo del controllo rappresenta la struttura, la forma, il modello per ciascuno degli elementi appartenenti. La classe è un progetto che determina quali dati e quali metodi avrà ogni istanza dell'insieme.

Ma ogni istanza (ogni elemento) è distinto, ovvero ciascuno ha il suo nome, la sua razza, il suo colore, il suo genere sessuale: distinto dagli altri.

In Visual Studio i controlli che si disegnano in un Form sono istanze di rispettive classi. Per esempio potrei avere due **`button`**, entrambi della classe **`Button`**, ma ciascuno di loro ha rispettive caratteristiche (nome, scritta, dimensione, posizione sul form, colore, ecc.). Analogamente accade per le **`TextBox`** e per le **`Label`** (e per qualsiasi altro controllo).




Nella figura qui a lato ci sono tre button, denominati rispettivamente button1, button2, button3.

Poiché tutti i button sono di tipo Button allora ciascun button possiede le caratteristiche (tra cui le proprietà) del button, tra cui per esempio le seguenti:

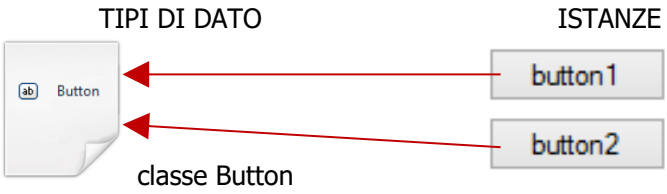
| Proprietà comuni ai button | Descrizione |
|---|---|
| `Text` | Testo visualizzato sul controllo |
| `BackColor` | Colore di sfondo del controllo |
| `Width` | Larghezza del controllo |
| `Height` | Altezza del controllo |
| `Visibile` | Indica se il controllo è visibile in esecuzione |
| `Enabled` | Indica se il controllo reagisce ai comandi dell'utente |

### IL CONCETTO DI TIPO

Il tipo è un modello di un elemento. Per esempio il tipo Button serve per definire le caratteristiche di qualsiasi button sia creato. Per comprendere intuitivamente questo concetto si può pensare al progetto di una casa, rappresentato su un foglio di carta: il progetto descrive una casa nei dettagli, ampiezza, numero di stanze, porte, finestre; tuttavia il progetto **NON** è una casa, ma solo un disegno; quindi non ci posso abitare. Da quel progetto però posso costruire tante case, tutte secondo lo stesso progetto: quindi ho molte case (distinte) dello stesso progetto (tipo).

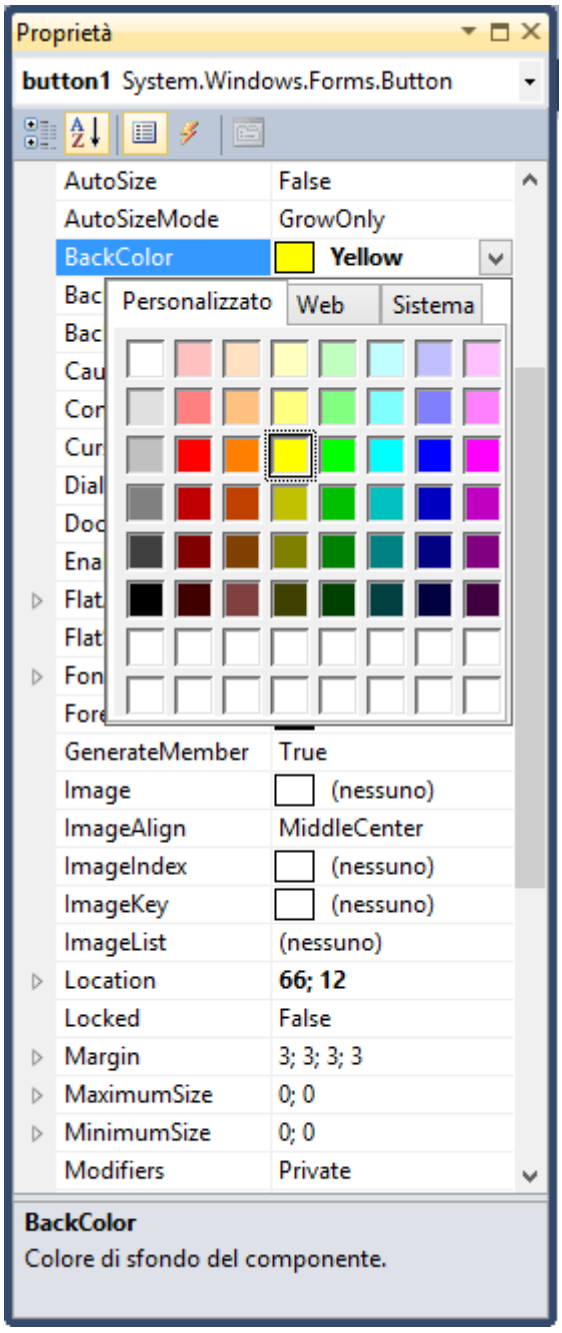
Il tipo determina come il controllo è fatto: l'aspetto, le capacità, il modo di interagire, ecc. ma ciascun controllo è distinto dagli altri, sebbene si somiglino.

Quando si crea un oggetto di un certo tipo, l'oggetto possederà automaticamente tutte le caratteristiche definite dal suo tipo.




Ogni button appartiene al tipo Button e ne possiede le caratteristiche.

## MODIFICARE LE PROPRIETÀ A TEMPO DI PROGETTAZIONE




Quando si sta creando un progetto Visual Studio è possibile cambiare l'aspetto dei controlli mediante le ripetitive Proprietà; in molti casi la modifica delle proprietà agisce immediatamente anche prima di mandarlo in esecuzione. Per esempio la modifica della proprietà Text fa cambiare subito il testo visualizzato anche nella finestra in via di progettazione.

In altri casi la modifica non si vede subito, ma solo quando si lancia in esecuzione il programma; per esempio se si imposta la proprietà Visible col valore False, questa modifica non rende subito invisibile il controllo, ma quando lo si lancia in esecuzione il controllo sarà invisibile.

Alcune proprietà sono di tipo numerico, che significa che è possibile impostare solo numeri come valori (per es. Width, la larghezza).

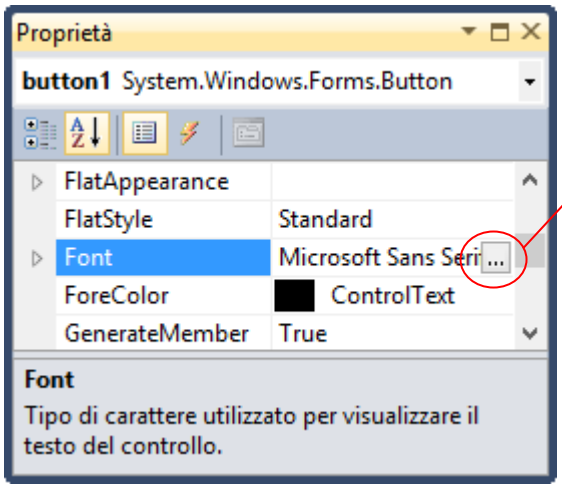
Alcune proprietà sono di tipo testuale, che significa che è possibile impostare solo frasi di caratteri come valori (per es. Text, il titolo).

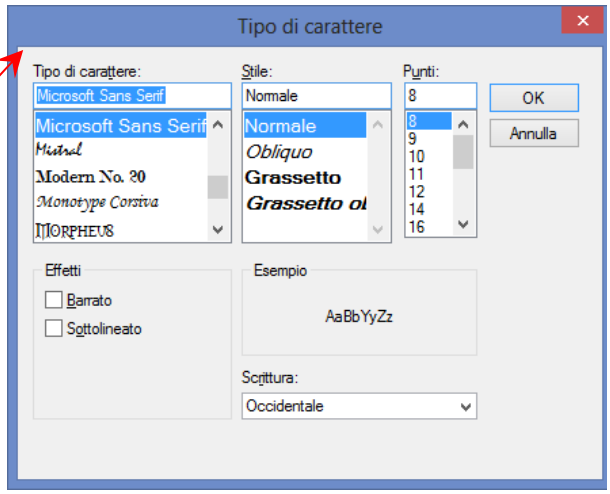
Alcune proprietà sono di tipo logico, che significa che è possibile impostare solo i valori true / false che significa vero / falso (per es. Visible, la visibilità).

Alcune proprietà sono di tipo articolato, che dipende dal tipo usato; per esempio la proprietà BackColor è di tipo Color ed ammette molti colori predefiniti, tra cui quelli visualizzati in un'apposita interfaccia.

Un altro esempio di proprietà di tipo articolato è quella con un pulsante detto ellisse che permette di aprire una finestra di dialogo specifica, come la proprietà Font che determina il carattere del controllo.

Come suggerimento di lavoro, crea un nuovo progetto e modifica tutte le proprietà sopra discusse, per comprendere l'azione.



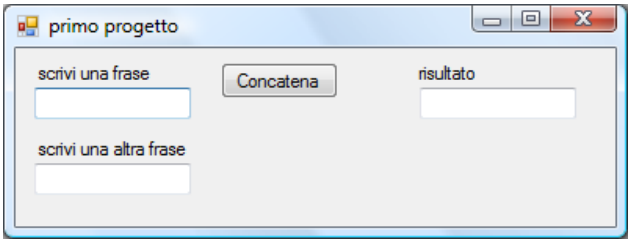




## GESTORI DI EVENTO

Un evento è un fatto che accade, che si verifica in un certo momento. In Windows ci sono molti eventi, tra cui, per esempio, il clic del mouse, la digitazione su tastiera o il clock del timer. In Visual Studio è possibile intercettare l'evento e gestirlo da programma. Vediamo come.

## PROGETTO GUIDATO 1.

- Prepara un form1 simile alla figura, con tre label, tre textBox e un pulsante




- Provalo (manda in esecuzione ed usa i controlli)
- Infine termina l'applicazione; ritorniamo in fase di progettazione e seleziona il pulsante
- Adesso fai DOPPIO CLIC sul pulsante
- L'ambiente si modifica e appare una finestra che mostra una pagina simile alla seguente:

VC#

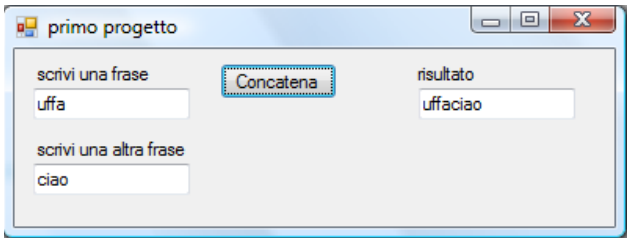
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
       private void button1_Click(object sender, EventArgs e)
        {
            |
        }
    }
}
```

- Nella posizione indicata dalla freccia lampeggia un cursore di inserimento testo
- Puoi scrivere in quel punto la seguente riga:

VC#

```
            textBox3.Text = textBox1.Text + textBox2.Text ;
```

- Nota che la riga deve stare in mezzo alle due parentesi graffe **{ … }** una sopra e una sotto.
- Poi manda in esecuzione (RUN) e usa il progetto come nella figura seguente:




- puoi scrivere nelle due caselle di testo a sinistra e premere il pulsante per visualizzare il risultato nella casella a destra
- quando hai finito termina l'esecuzione; salva tutto e chiudi

## ISTRUZIONI E GESTORE DI EVENTO

Quando in fase di progettazione si fa doppio clic su un controllo, il RAD prepara un pezzo di programma detto gestore di evento; in particolare il doppio clic sul pulsante predispone il gestore dell'evento clic per il pulsante stesso.

Il gestore di evento è un frammento di codice (di programma) che rappresenta il modo con cui il linguaggio deve gestire un determinato evento (accadimento):

VC#
```
private void button1_Click(object sender, EventArgs e)
{
        |
}
```

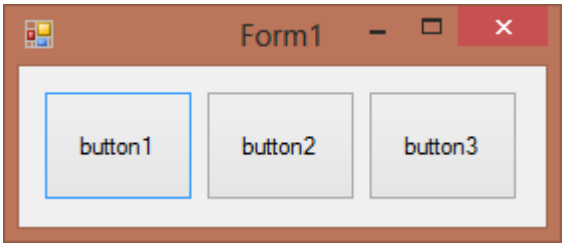
- La scritta **`private void`** non ci interessa per il momento; se ne discuterà in seguito.
- La scritta **`button1_Click`** è il nome del gestore di evento; serve per distinguerlo dagli altri e, in questo caso, significa gestione del Click sul button1.
- La scritta **`(object sender, EventArgs e)`** non ci interessa per il momento; tra parentesi tonde ci sono i cosiddetti «**parametri**» del metodo ma se ne discuterà più in là.
- Le parentesi graffe delimitano il «**corpo del metodo**» ed indicano la posizione in cui è possibile scrivere le istruzioni per gestire l'evento: è quello per noi più importante.

Quindi per questa prima fase di apprendimento del linguaggio ci possiamo limitare alla sola parte racchiusa tra parentesi graffe: lo chiameremo il **corpo del metodo**.

Tra le parentesi graffe è possibile scrivere dei comandi che dicono quale comportamento deve assumere il programma, quando si verifica l'evento previsto (il clic del mouse sul button1).

## PROGETTO GUIDATO 2.

- Prepara un form1 simile alla figura, con tre pulsanti




- Fai doppio clic sul button1 e attendi che l'ambiente prepari la finestra col gestore di evento **`button1_Click`**
- Scrivi la seguente istruzione nel punto dove lampeggia il cursore, proprio in messo alle parentesi graffe (osserva che tutto il resto non devi modificarlo):

VC#
```
    BackColor = Color.Aqua;
```

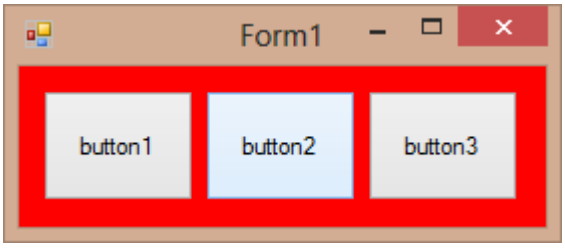
- Torna al From e fai doppio clic sul button2 e attendi che l'ambiente prepari la finestra col gestore di evento **`button2_Click`**
- Scrivi la seguente istruzione nel punto dove lampeggia il cursore, proprio in messo alle parentesi graffe (osserva che tutto il resto non devi modificarlo):

VC#
```
    BackColor = Color.Red;
```

- Torna al From e fai doppio clic sul button3 e attendi che l'ambiente prepari la finestra col gestore di evento **button3_Click**
- Scrivi la seguente istruzione nel punto dove lampeggia il cursore, proprio in messo alle parentesi graffe (osserva che tutto il resto non devi modificarlo):

VC#
```
BackColor = Color.Yellow;
```

- Poi manda in esecuzione (RUN) e usa il progetto cliccando sui pulsanti alternativamente:




- Quando fai clic su button1 viene eseguita l'istruzione scritta in **button1_Click**
- Quando fai clic su button2 viene eseguita l'istruzione scritta in **button2_Click**
- Quando fai clic su button3 viene eseguita l'istruzione scritta in **button3_Click**

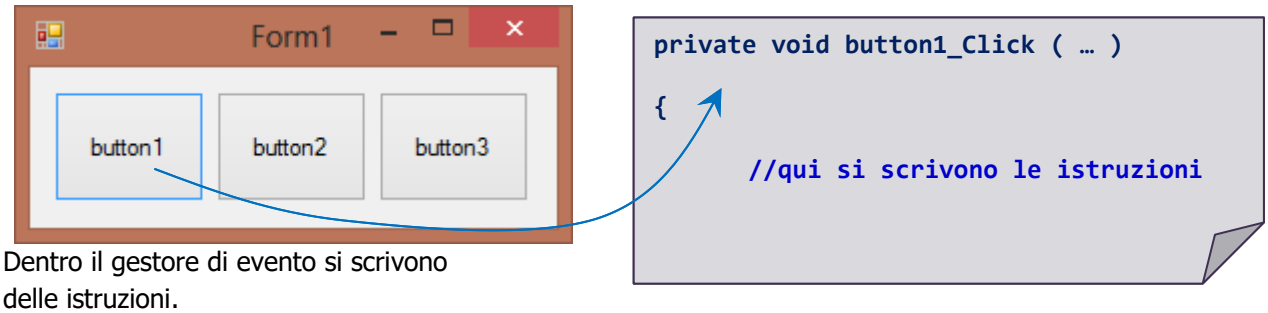
## MODIFICARE LE PROPRIETÀ A RUNTIME

### ISTRUZIONI E PROPRIETÀ

Un gestore di evento è un frammento di programma (un insieme di istruzioni che perseguono uno scopo) che serve per stabilire come deve comportarsi l'applicazione quando si verifica un certo evento.

Quando si desidera che ad un dato evento (per es. il clic del mouse) l'applicazione reagisca con un comportamento stabilito si deve agganciare l'evento legato ad un controllo con un suo gestore di evento.

Per legare il **button1** ad un gestore d'evento **button1_clic** è sufficiente fare doppio clic sul controllo in fase di progettazione:




Dentro il gestore di evento si scrivono delle istruzioni.

Un'istruzione è un comando che verrà eseguito e che causa un effetto (manifesto o occulto) ai dati del programma. Per far eseguire le istruzioni è necessario mandare in esecuzione l'applicazione. Se le istruzioni sono legate a un gestore di evento, allora è necessario che si sollevi l'evento previsto, come il clic del mouse. In un gestore di evento si possono scrivere molte istruzioni.

Quando il programma è in esecuzione e si verifica l'evento previsto (per es. il clic su un button1) le istruzioni presenti nel gestore di evento sono eseguite in sequenza, ovvero nell'ordine in cui sono scritte.

L'istruzione più semplice è l'assegnazione. L'assegnazione è l'istruzione che impone di porre un valore in una locazione. Nel caso delle proprietà, l'assegnazione permette di modificare il valore di una proprietà. Per esempio:

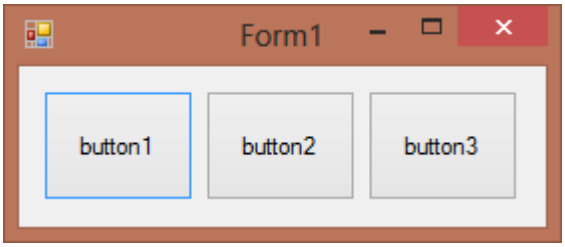
VC#
```
BackColor = Color.Yellow;
```

impone che la proprietà **BackColor** sia impostata col valore **Color.Yellow;** cioè deve diventare gialla.

Il concetto importante è quindi che è possibile modificare il valore di una proprietà anche a tempo di esecuzione, scrivendo un programma (detto codice) in un gestore di evento (per es. button1_clic) composto da tante istruzioni tra cui quelle di assegnazione.

Nel precedente esempio guidato, il programma costruito prevede di modificare il colore della finestra a seconda del pulsante premuto. Vediamo adesso un esempio per modificare i colori dei singoli controlli.

## PROGETTO GUIDATO 3.




- Prepara un form1 simile alla figura, con tre pulsanti
- Fai doppio clic sul button1 e attendi che l'ambiente prepari la finestra col gestore di evento **button1_Click**

- Scrivi le seguenti istruzioni nel punto dove lampeggia il cursore, proprio in mezzo alle parentesi graffe:

VC#
```
button1.BackColor = Color.Blue;
button3.BackColor = Color.Coral;
```

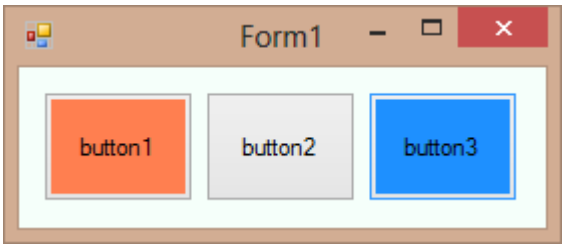
- Torna al From e fai doppio clic sul button2 e attendi che l'ambiente prepari la finestra col gestore di evento **button2_Click**
- Scrivi le seguenti istruzioni nel punto dove lampeggia il cursore, proprio in mezzo alle parentesi graffe:

VC#
```
button3.BackColor = Color.DodgerBlue;
BackColor = Color.MintCream;
```

- Torna al From e fai doppio clic sul button3 e attendi che l'ambiente prepari la finestra col gestore di evento **button3_Click**
- Scrivi le seguenti istruzioni nel punto dove lampeggia il cursore, proprio in mezzo alle parentesi graffe:

VC#
```
button1.BackColor = Color.OldLace;
BackColor = Color.Black;
```

- Poi manda in esecuzione (AVVIA/RUN) e usa il progetto cliccando sui pulsanti alternativamente:




- Quando fai clic su button1 viene eseguita l'istruzione scritta in **button1_Click**
- Quando fai clic su button2 viene eseguita l'istruzione scritta in **button2_Click**
- Quando fai clic su button3 viene eseguita l'istruzione scritta in **button3_Click**

## RIFLESSIONI SUL PROGETTO GUIDATO

Osserviamo il precedente programma.

Quando si fa clic sul pulsante button1 si innesca il gestore di evento `button1_Click` che fa eseguire le due istruzioni contenute tra le parentesi graffe. Le istruzioni modificano rispettivamente il colore del button1 medesimo e il colore di button3.

Quando si fa clic sul pulsante button2 si innesca il gestore di evento `button2_Click` che fa eseguire le due istruzioni contenute tra le parentesi graffe. Le istruzioni modificano rispettivamente il colore del button3 e il colore della finestra.

Quando si fa clic sul pulsante button3 si innesca il gestore di evento `button3_Click` che fa eseguire le due istruzioni contenute tra le parentesi graffe. Le istruzioni modificano rispettivamente il colore del button1 e di nuovo il colore della finestra.

Dal programma si deduce che è possibile digitare più istruzioni dentro il gestore di evento (nel nostro caso due) e che vengono eseguite in sequenza (nell'ordine in cui sono scritte).

Un altro aspetto interessante riguarda il destinatario dell'istruzione di assegnazione. Per esempio l'istruzione:

```
VC#
button1.BackColor = Color.Blue;
```

modifica il colore del button1 ovvero quello specificato nella scritta a sinistra dell'istruzione. Invece l'istruzione:

```
VC#
button3.BackColor = Color.Coral;
```

modifica il colore del button3 come specificato nella scritta a sinistra dell'istruzione.

Infine l'istruzione:

```
VC#
BackColor = Color.Black;
```

modifica il colore della finestra.

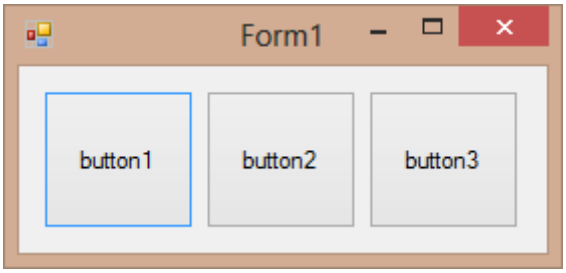
In sintesi si deve seguire la seguente regola generale:

- Se non si specifica un controllo coinvolto, la proprietà si riferisce all'intera finestra;
- Se si specifica un controllo determinato, la proprietà si riferisce al controllo;

Vediamo adesso altri esempi che modificano altre proprietà dei controlli e della finestra.

## PROGETTO GUIDATO (TEXT)

- Prepara un form1 simile alla figura, con tre pulsanti, tre etichette e tre caselle di testo




- Adesso fai DOPPIO CLIC sul **button1** e scrivi le seguenti istruzioni nel corpo:

VC#
```
Top = 100;
button1.Top = 10;
button2.Top = 50;
button3.Top = 100;
```

- Osserva che ci sono tre righe diverse (tre istruzioni separate) tutte all'interno delle parentesi graffe del gestore di evento
- Prova a usare il programma: cosa accade premendo il pulsante?
- Infine termina l'applicazione; ritorniamo in fase di progettazione e seleziona il pulsante **button2** e fai DOPPIO CLIC e scrivi le seguenti istruzioni nel corpo:

VC#
```
Text = "Mia";
button1.Text = "uno";
button2.Text = "due";
button3.Text = "tre";
```

- Seleziona il pulsante **button3** e fai DOPPIO CLIC e scrivi le seguenti istruzioni:

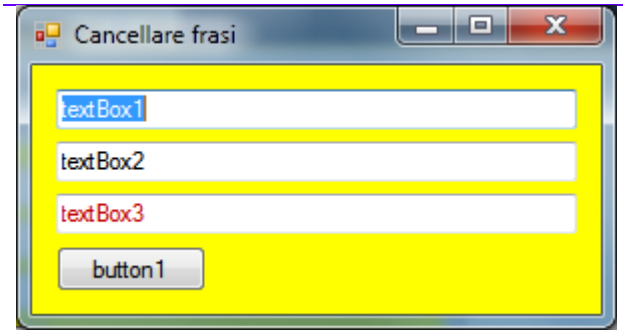
VC#
```
Text = Text + "o";
button1.Text = button1.Text + "1";
button2.Text = button2.Text + "2";
button3.Text = button3.Text + "3";
```

## RIFLESSIONI SUL PROGETTO GUIDATO

Text è la proprietà che si riferisce al testo mostrato da un controllo. Per la finestra la proprietà si riferisce al titolo visualizzato sulla barra del titolo; per un pulsante si riferisce al testo mostrato sopra il pulsante.

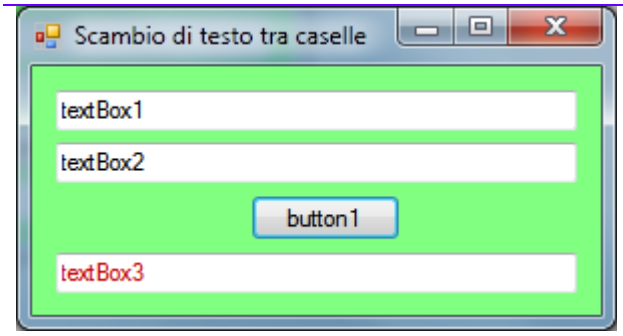
# ESERCIZI

## FRASI BASE



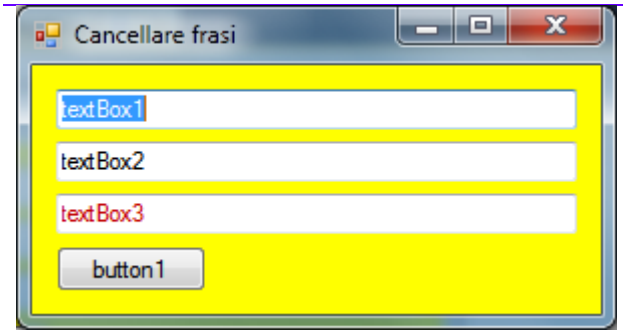

- Scrivere il programma con un pulsante button1 che, quando premuto, scrive nelle caselle di testo le seguenti frasi:
  - «Amadeus»
  - «Busillis»
  - «Claudicò»
- Provare il programma

## SCAMBIARE FRASI



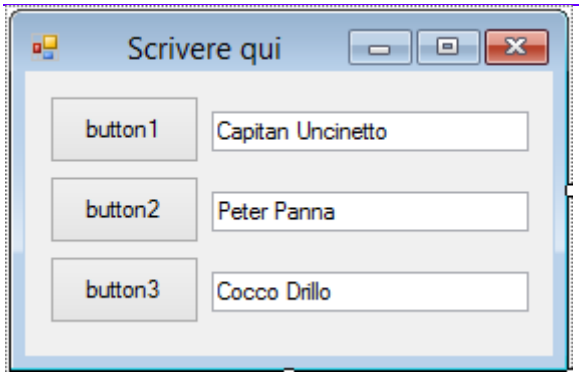

- Scrivere il programma con un pulsante button1 che, quando premuto:
  - copia il testo della prima casella nella terza;
  - Poi copia il testo della seconda casella nella prima;
  - Poi copia il testo della terza casella nella prima;
- Provare il programma

## NUMERI E STRINGHE




- Scrivere il programma con un pulsante button1 che, quando premuto, scrive nelle caselle di testo i seguenti numeri:
  - 121
  - 232
  - 343
- Provare il programma

## CAMBIARE IL TITOLO




- Scrivere il programma che fa comparire nel titolo della finestra il testo della casella di testo corrispondente:
- Il pulsante button1 copia il testo dalla textBox1 al titolo della finestra
- Il pulsante button2 copia il testo dalla textBox2 al titolo della finestra
- Il pulsante button3 copia il testo dalla textBox3 al titolo della finestra

- Si osservi che l'utente può scrivere quello che desidera nella casella desiderata e poi col pulsante lo copia nel titolo della finestra; quindi le frasi proposte sono arbitrarie!

# SOMMARIO