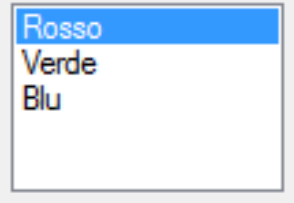
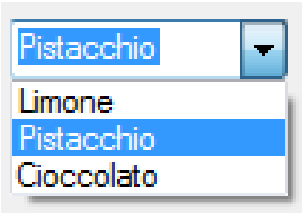
ListBox e ComboBox

Prof. Francesco Accarino
IIS Altiero Spinelli Sesto San Giovanni

GESTIRE COLLEZIONI DI DATI





Due tipi di controllo, ListBox e ComboBox, consentono di gestire collezioni di dati unidimensionali. Tali controlli consentono di:

- ❑ visualizzare una lista di valori di varia natura (numerica, testuale, eccetera);
- ❑ aggiungere e togliere un valore o un insieme di valori;
- ❑ cancellare tutti i valori della lista;
- ❑ "popolare" la lista di valori mediante l'assegnazione di un vettore.
- ❑ eseguire ricerche;
- ❑ mantenere la lista ordinata;

Entrambi i tipi di controllo condividono queste e altre caratteristiche. Il tipo ComboBox, inoltre, aggiunge a queste le funzionalità di un TextBox, e dunque "combina" le caratteristiche di due tipi di controlli (ComboBox sta appunto per «Casella combinata»).

CLASSE «ListBox»

Proprietà della classe ListBox.

PROPRIETÀ – TIPO	DESCRIZIONE
DataSource `object`	Assegnando una oggetto vettore a questa proprietà è possibile, con una sola istruzione, popolare il `ListBox` con i valori contenuti nel vettore.
Items `object[]`	`Items` è la proprietà attraverso la quale si può accedere agli elementi della lista. Essa espone a sua volta proprietà e metodi attraverso i quali aggiungere e togliere elementi, conoscere il numero degli elementi, eccetera: `Items.Count`: memorizza il numero degli elementi della lista; `Items.Add(object)`: aggiunge un elemento alla lista; `Items.AddRange(object[])`: aggiunge una sequenza di elementi; `Items.RemoveAt(int)`: rimuove un elemento dalla lista; `Items.Clear()`: rimuove tutti gli elementi dalla lista.
SelectedIndex `int`	Indice dell'elemento correntemente selezionato. (Tale elemento viene di norma visualizzato in bianco su blu.) Se è non selezionato alcun elemento, `SelectedIndex` vale −1.
SelectedItem `object`	Riferimento all'elemento correntemente selezionato. Il riferimento è di tipo `object`, dunque, perché possa essere assegnato a una variabile, occorre che sia eseguita l'appropriata operazione di cast.
SelectionMode `SelectionMode`	Definisce il tipo di selezione ammissibile: `None`: non è possibile selezionare alcun elemento; `One`: è possibile selezionare un solo elemento per volta; `MultiSimple`: è possibile selezionare più elementi; `MultiExtended`: è possibile selezionare più elementi; l'utente può usare i tasti CTRL e SHIFT e tasti freccia per eseguire la selezione.
Sorted `bool`	Indica se la lista debba o meno essere mantenuta ordinata. Il valore di default è `false` (lista non ordinata).

CLASSE «ListBox»

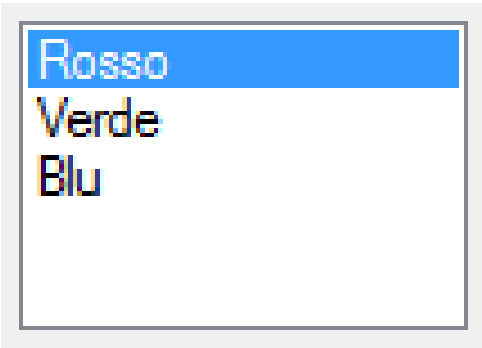
Del nutrito insieme di eventi pubblicato dalla classe ListBox vengono comunemente gestiti quelli che rispondono alla selezione di un elemento da parte dell'utente.

EVENTO	DESCRIZIONE
`Click e DoubleClick`	Entrambi vengono sollevati dopo che l'utente ha cliccato (o eseguito un doppio clic) sull'area del `ListBox`. Se il clic (o il doppio clic) avviene su un elemento questo diventa selezionato prima che l'evento sia sollevato.
`SelectedIndexChanged`	Viene sollevato dopo che è stato selezionato un elemento diverso da quello corrente.

Popolare un «ListBox» attraverso l metodi «Add()» e «AddRange()»

```
lboColori.Items.Add("Rosso");
lboColori.Items.Add("Verde");
lboColori.Items.Add("Blu");
```

```
string[ ] colori = {"Rosso" , "Verde" , "Blu"};
lboColori.Items.AddRange(colori);
```



Popolare un «ListBox» attraverso la proprietà «DataSource»

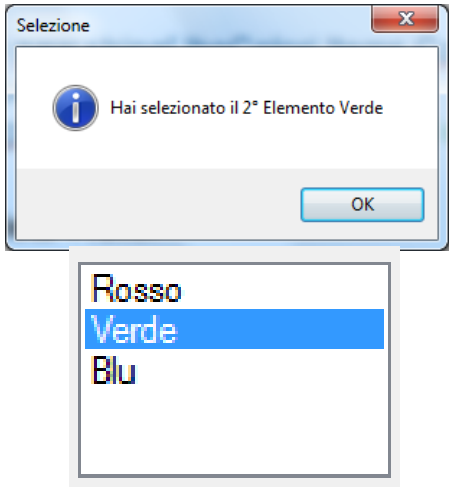
```
string[ ] colori = {"Rosso" , "Verde" , "Blu"};
lboColori. DataSource = colori ;
```

Accesso agli elementi di un «ListBox»

```
string[ ] sColori = new string[ lboColori.Items.Count ];
for (int i = 0; i < lboColori.Items.Count; i++)
sColori[i] = (string) lboColori.Items[i]; // uso del cast
```

Uso della proprietà «Text»

Anche il controllo ListBox definisce la proprietà Text, della quale fa un uso non completamente convenzionale. Mediante essa è possibile ottenere una rappresentazione stringa dell'elemento attualmente selezionato.
In alcune situazioni, dunque, può essere usata come sostituto della proprietà SelectedItem.

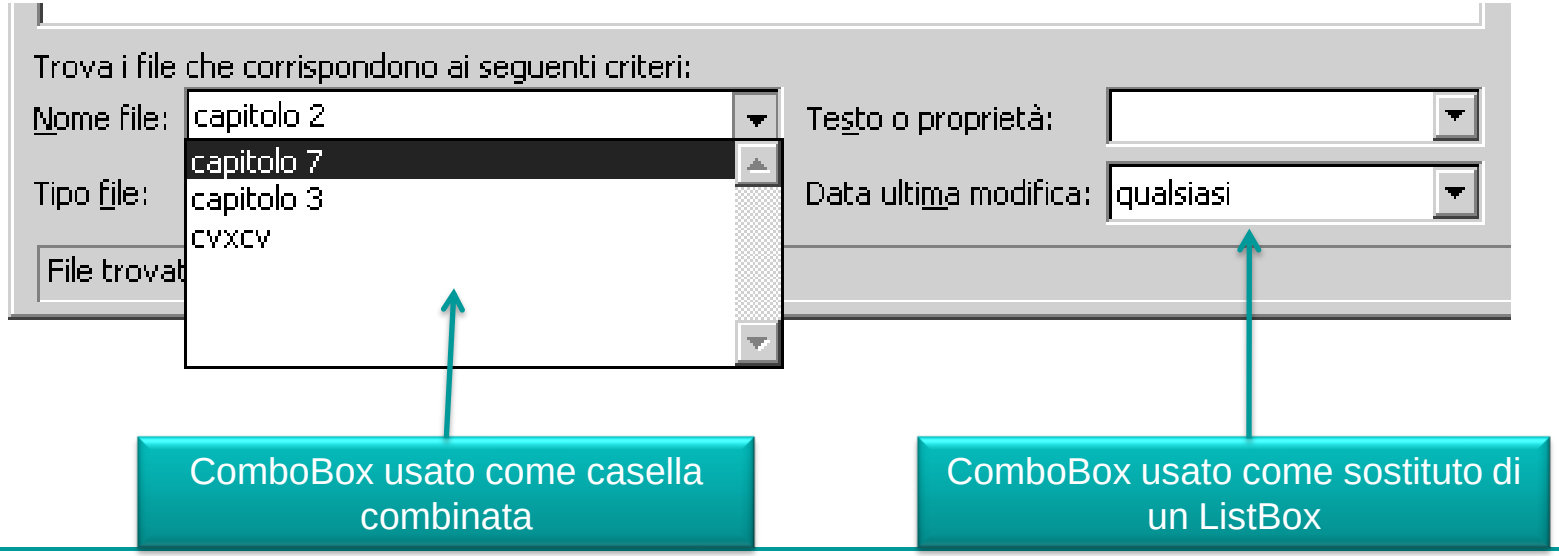



```
MessageBox.Show("Hai selezionato il " + (lboColori.SelectedIndex + 1).ToString() + "° Elemento "
                + lboColori.Text, "Selezione", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

CLASSE «ComboBox»

I ComboBox combinano molte caratteristiche di un ListBox con quelle di un TextBox. Il controllo può dunque comunicare con l'utente sia attraverso il mouse (caratteristica tipica del ListBox) sia attraverso la tastiera (caratteristica tipica del TextBox). Tipicamente, l'utente può:

- cliccare sulla freccia per visualizzare «l'elenco a discesa» (la lista degli elementi) e quindi selezionare un elemento; l'elemento selezionato viene visualizzato nella casella;
- digitare all'interno della casella.



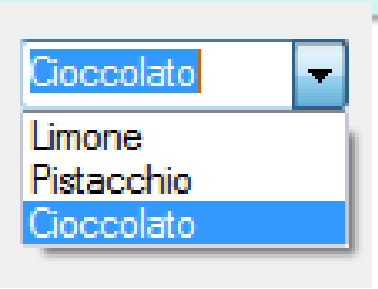
Proprietà della classe ComboBox.

PROPRIETÀ - TIPO	DESCRIZIONE
DropDownStyle `ComboBoxStyle`	Questa proprietà caratterizza il comportamento del `ComboBox` e dunque la particolare funzione per il quale viene inserito nell'interfaccia. Possibili valori sono: `Simple`: L'utente può digitare nella casella oppure selezionare un elemento dall'elenco a discesa, il quale è sempre visibile. (Dunque non è presente la freccia per visualizzarlo). `DropDown`: L'utente può digitare nella casella oppure selezionare un elemento dall'elenco a discesa, il quale viene visualizzato cliccando sulla freccia. `DropDownList`: L'utente non può digitare nella casella e dunque può soltanto selezionare un elemento dall'elenco a discesa, il quale viene visualizzato cliccando sulla freccia.
MaxDropDownItems `int`	Massimo numero di elementi visualizzati contemporaneamente nell'elenco a discesa.
MaxLength e **Text**	Entrambe svolgono la stessa funzione delle proprietà omologhe esposte dalla classe `TextBox`.

Eventi

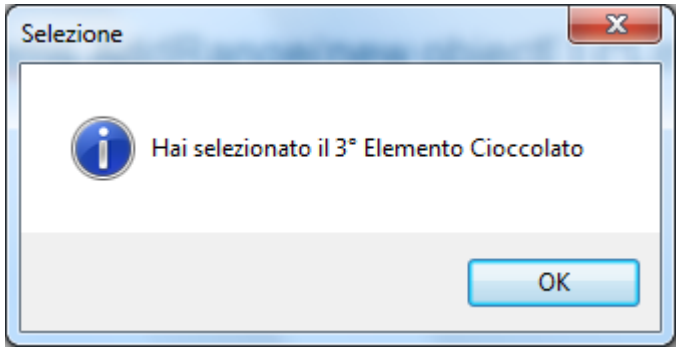
Gli eventi pubblicati dalla classe ComboBox sono sostanzialmente gli stessi delle classi ListBox e TextBox. Quelli gestiti più comunemente sono: TextChanged, Click, DoubleClick, SelectedIndexChanged.

Uso di una ComboBox

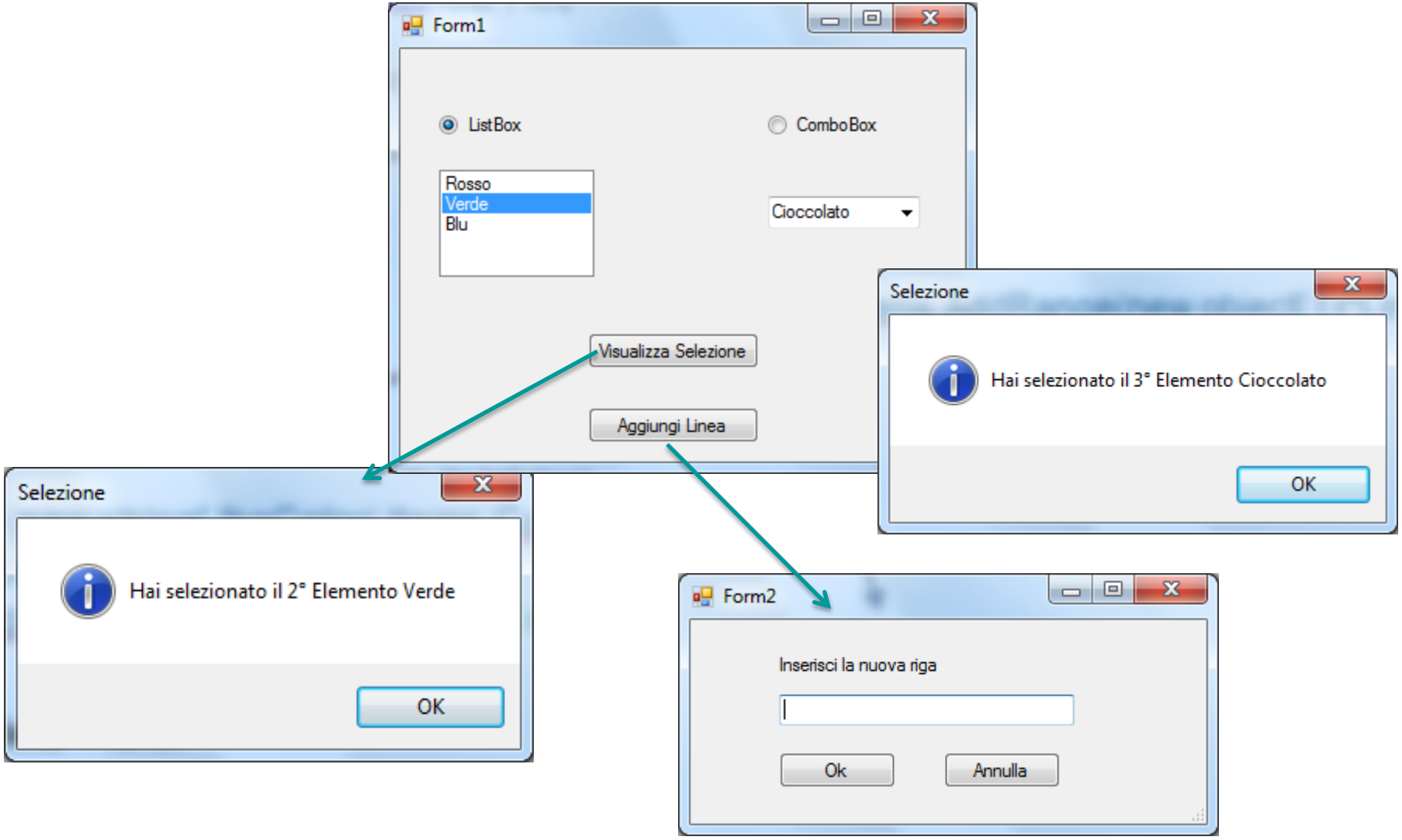



```
this.cboGusti.Items.AddRange(new object[ ] {"Limone“, "Pistacchio“,"Cioccolato"});
```

```
MessageBox.Show("Hai selezionato il " + (cboBusti.SelectedIndex + 1).ToString() + "° Elemento "
                + cboGusti.Text, "Selezione", MessageBoxButtons.OK, MessageBoxIcon.Information);
```



Esercizio: Realizzare un'applicazione che si comporta come quella mostrata nella figura seguente:



Suggerimenti

```
public partial class Form2 : Form
{
    Form1 f;
    public Form2(Form1 f)
    {
        this.f = f;
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (f.rbuLista.Checked == true)
        {
            if (NuovaLineaTXT.Text != "")
                f.lboColori.Items.Add(NuovaLineaTXT.Text);
        }
        else
        {
            if (NuovaLineaTXT.Text != "")
                f.cboGusti.Items.Add(NuovaLineaTXT.Text);
        }

        this.Close();
    }

    private void button2_Click(object sender, EventArgs e)
    {

        this.Close();
    }
}
```

Modificare la classe Form2 per fare in modo che il costruttore riceva un oggetto di tipo Form1 in modo tale da poter accedere alle sue componenti e modificarle.

Suggerimenti

Nel file Form1.cs inserire il codice seguente come gestore dell'evento Load del Foem per l'inizializzazione delle componenti

```
private void Form1_Load(object sender, EventArgs e)
        {
            cboGusti.SelectedIndex = 2;
            lboColori.SelectedIndex = 0;
            rbuLista.Checked = true;
        }
```

Nel file Form1.Designer.cs aggiungere la linea seguente per aggiungere l'ascolto dell'evento Load e specificare come gestore di questo evento la funzione che abbiamo scritto

```
this.Load += new System.EventHandler(this.Form1_Load);
```