



# 10. Dialoghi, menu e altre meraviglie

**Un esempio di applicazione con la presentazione di alcuni elementi indispensabili in quasi tutti i programmi: le finestre comuni di dialogo, la stampa, i menu e la barra di stato.**

In Visual Basic 6.0 esisteva un unico controllo in grado di visualizzare la finestra di dialogo desiderata, tra quelle disponibili, impostandone opportunamente le proprietà.

In Visual Studio 2008 esistono invece diversi controlli specializzati, uno per ogni tipo di finestra di dialogo. Vedremo dettagliatamente ognuno di questi controlli, così come approfondiremo l'uso dei controlli per gestire la stampa e l'anteprima di stampa. Infine, vedremo anche come si creano e gestiscono i menu e la barra di stato.

Inseriremo tutti questi esempi in un'applicazione che, al suo completamento, sarà un vero e proprio editor di file di testo perfettamente funzionante.

Questi controlli sono disponibili in tutte le edizioni di Visual Basic e Visual Studio 2008.

## Le finestre comuni di dialogo

Le operazioni che vengono eseguite più frequentemente da un'applicazione sono le seguenti (tra parentesi è indicato il nome del controllo dedicato):

- ❖ aprire un file (OpenFileDialog);
- ❖ salvare un file (SaveFileDialog);
- ❖ selezionare una cartella (FolderBrowserDialog);
- ❖ selezionare un colore (ColorDialog);
- ❖ selezionare un font (FontDialog).

Esistono, poi, tre ulteriori controlli che ci permettono di gestire la stampa di un documento. Vedremo questi controlli più avanti nel presente capitolo.

### OpenFileDialog

Esistono molti casi in cui abbiamo la necessità di selezionare un file da aprire: un file di testo (.txt), un documento di Office (.doc, .xls, .ppt, .mdb), un'immagine, una pagina web salvata in locale (.html, .htm), un file musicale (.wav, .mp3), un video (.avi, .mpeg) o altro ancora.

Il controllo OpenFileDialog permette di selezionare il nome del file da aprire e di eseguire anche l'apertura vera e propria del file (Figura 10.1).

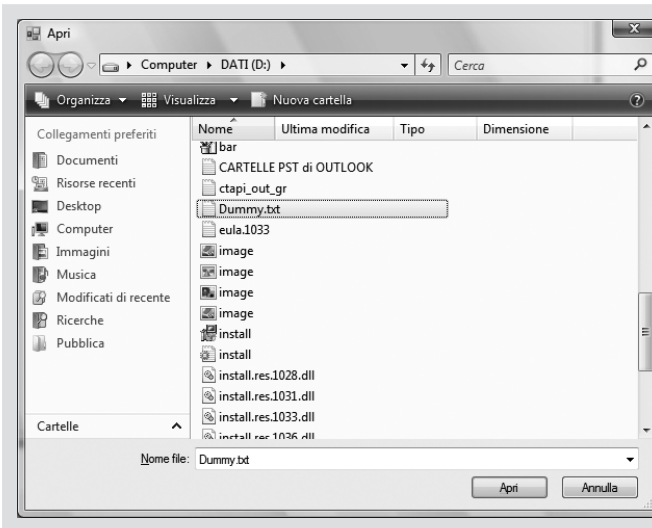


Figura 10.1 - La finestra di dialogo OpenFileDialog.

Le proprietà più comunemente utilizzate di questo controllo sono le seguenti:

- ❖ `InitialDirectory`: imposta il percorso completo della cartella iniziale che sarà selezionata dal controllo;
- ❖ `Filter`: elenca uno o più filtri che compariranno nella casella combinata **Tipo file** della finestra di dialogo. I filtri sono formati da coppie descrizione-estensione di file, separate da una barra verticale (|) sia tra di loro sia tra una coppia e l'altra. Il seguente esempio mostra lo schema di impostazione di un filtro: `"documenti di Word (*.doc) | *.doc | file di testo (*.txt) | *.txt | tutti i file (*.*) | *.*"`;
- ❖ `FilterIndex`: è un numero intero che indica quale, tra i filtri impostati con la proprietà `Filter`, debba essere quello selezionato. Il valore di default è 1.
- ❖ `RestoreDirectory`: imposta un valore booleano (`True` o `False`) che indica se la cartella corrente viene ripristinata dopo la chiusura del controllo;
- ❖ `FileName`: imposta o legge il nome del file selezionato con il controllo. Il nome comprende anche il percorso completo.

I metodi più utilizzati, invece, sono i seguenti:

- ❖ `ShowDialog`: visualizza la finestra di dialogo per la selezione del file da aprire;
- ❖ `OpenFile`: apre un flusso (`Stream`) e restituisce il riferimento al flusso stesso.

Vediamo un esempio di applicazione per leggere il contenuto di un file:

1. create un nuovo progetto;
2. inserite due pulsanti (`Button`), lasciando il loro nome predefinito, quindi modificate la loro proprietà `Text` rispettivamente con le descrizioni `Leggi file` e `Cancella testo`;
3. inserite una casella di testo (`TextBox`), impostate la proprietà `Multiline` a `True` (oppure fate clic sulla casella di spunta del suo `SmartTag`) e ridimensionate la casella fino quasi a riempire l'intero form;

- aggiungete un controllo `OpenFileDialog` con un doppio clic sulla voce corrispondente nel gruppo **Finestre di dialogo (Dialogs)** della **Casella degli Strumenti (Toolbox)**, lasciando il nome di default;
- fate doppio clic su un pulsante o sul form e sostituite tutto il codice con il seguente:

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender _
        As System.Object, _
        ByVal e As System.EventArgs) _
        Handles Button1.Click

        OpenFileDialog1.InitialDirectory = "C:\\"
        OpenFileDialog1.Filter = _
            "file di testo (*.txt)|*.txt|" & _
            "tutti i file (*.*)|*.*"
        OpenFileDialog1.FilterIndex = 1
        openFileDialog1.RestoreDirectory = True

        If openFileDialog1.ShowDialog() = _
            Windows.Forms.DialogResult.OK Then
            ` codice per leggere il file
            Me.TextBox1.Text = _
                My.Computer.FileSystem.ReadAllText(" " & _
                    OpenFileDialog1.FileName)
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender _
        As System.Object, _
        ByVal e As System.EventArgs) _
        Handles Button2.Click
        Me.TextBox1.Text = ""
    End Sub
End Class
```



Il codice sopra riportato include il codice relativo ai gestori degli eventi `Click` sia del pulsante `Button1` sia del pulsante `Button2`.

- premete il pulsante **F5** per avviare il programma;
- fate clic sul pulsante **Leggi file**: apparirà una finestra per l'apertura di un file, posizionata alla cartella `C:\` e con il filtro preimpostato a

**file di testo (\*.txt)**, ma con la possibilità di scegliere anche **tutti i file (\*.\*)**;

8. selezionate un file di testo qualsiasi e premete il pulsante **Apri**: la finestra scomparirà e il contenuto del file di testo selezionato verrà copiato nella casella di testo del form della nostra applicazione.

L'applicazione, a questo punto, avrà un aspetto simile a quello rappresentato in Figura 10.2.

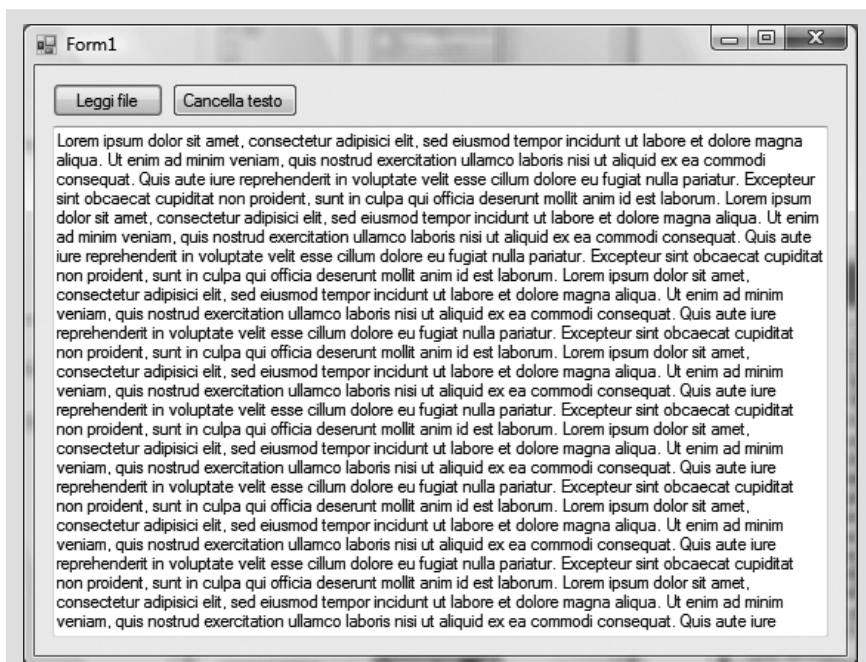


Figura 10.2 - Applicazione di esempio per OpenFileDialog.



Il file di testo selezionato non rimane aperto: il metodo *My.Computer.FileSystem.ReadAllText* apre il file, legge il contenuto e poi lo richiude. Le modifiche apportate al testo contenuto nella casella di testo, quindi, non si ripercuotono nel contenuto del file salvato su disco.

Dato che abbiamo inserito anche il codice per cancellare il contenuto della casella di testo, possiamo svuotarla semplicemente premendo il pulsante `Cancella testo`.



L'applicazione mostrata, nel suo piccolo, vi può dare già una rudimentale capacità di archiviare i vostri testi. Se, per esempio, registrate le vostre ricette in file di testo (un file per ogni ricetta) e le archiviate in una cartella, con questa applicazione sarete in grado di selezionare una ricetta e di visualizzarla sullo schermo in modo estremamente semplice. Per una migliore organizzazione potreste perfino creare cartelle diverse per le diverse tipologie di piatti (primi, secondi, contorni, dolci ecc.), così da ritrovare più facilmente la ricetta che vi interessa.

## SaveFileDialog

L'applicazione che abbiamo appena mostrato ha un piccolo-grande difetto: può solo leggere un file di testo ma non registra le eventuali modifiche che voi potreste apportare al contenuto della casella di testo.

Possiamo ovviare a questo problema apportando due modifiche all'esempio:

- ❖ l'utilizzo del controllo **SaveFileDialog** per selezionare il nome del file, completo di percorso, da utilizzare per salvare il testo (Figura 10.3);
- ❖ l'utilizzo di un particolare metodo per salvare il testo, cioè `My.Computer.FileSystem.WriteAllText`.

Aperte nuovamente l'esempio utilizzato per illustrare l'uso del controllo `OpenFileDialog` e apportate le seguenti modifiche:

1. inserite un nuovo pulsante e modificate la proprietà `Text` in `Salva file`;
2. inserite un controllo **SaveFileDialog** dal gruppo **Finestre di dialogo (Dialogs)** della **Casella degli strumenti (Toolbox)**;

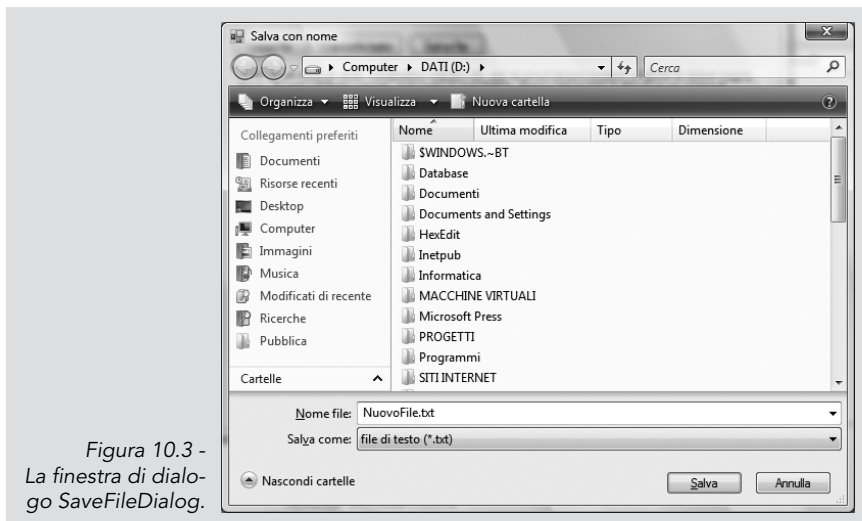


Figura 10.3 -  
La finestra di dialogo  
SaveFileDialog.

3. inserite il seguente codice subito dopo l'evento Click di Button1:

```
Private Sub Button3_Click(ByVal sender _
    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles Button3.Click
    SaveFileDialog1.InitialDirectory = "d:\\"
    SaveFileDialog1.Filter = _
        "file di testo (*.txt)|*.txt|" & _
        "tutti i file (*.*)|*.*"
    SaveFileDialog1.FilterIndex = 2
    SaveFileDialog1.RestoreDirectory = True

    If SaveFileDialog1.ShowDialog() = _
        Windows.Forms.DialogResult.OK Then
        My.Computer.FileSystem.WriteAllText(" " & _
            SaveFileDialog1.FileName, _
            Me.TextBox1.Text, False)
    End If
End Sub
```

4. lanciate il programma con il tasto **F5**.

La situazione sarà identica a quella dell'esempio precedente (Figura 10.4),  
tranne per il fatto che ora abbiamo la possibilità di salvare il testo presente  
nella casella di testo.

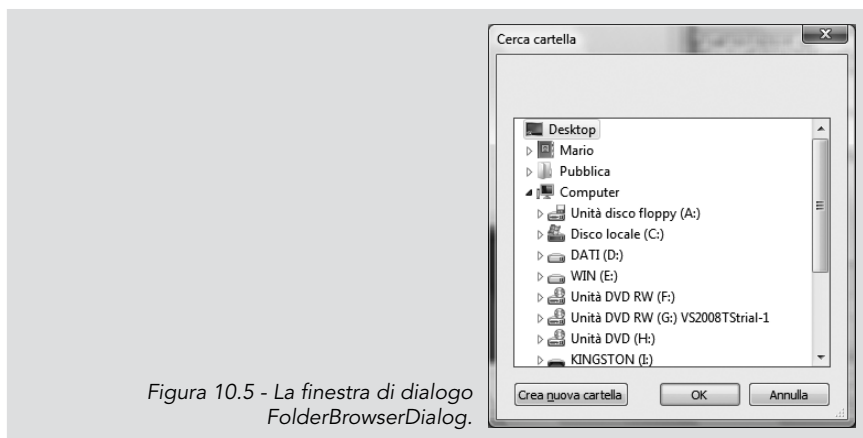




Adesso avete non solo la possibilità di leggere le ricette già registrate sul file, ma potete anche modificarle e creare dei file con le nuove ricette!

## FolderBrowserDialog

Aggiungendo un controllo di questo tipo possiamo selezionare una cartella esistente o perfino crearne una nuova nella posizione desiderata (Figura 10.5).



Riprendete ancora una volta l'esempio precedente e modificalo come segue:

1. aggiungete un controllo FolderBrowserDialog;
2. aggiungete un nuovo pulsante, impostando la proprietà Text a Seleziona cartella;
3. aggiungete una nuova casella di testo (TextBox2) e posizionala tra i pulsanti e la casella di testo già presente, ridimensionandola (Figura 10.6);

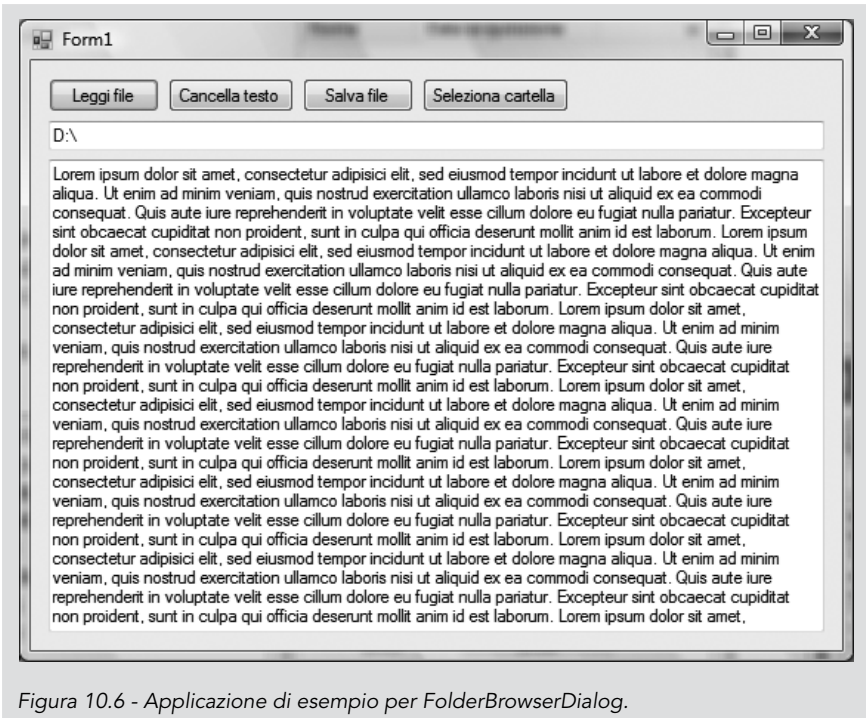


Figura 10.6 - Applicazione di esempio per FolderBrowserDialog.

4. impostate la proprietà Text della nuova casella di testo a "C:\\";
5. inserite il seguente codice sotto quello relativo all'evento Click di Button3:

```
Private Sub Button4_Click(ByVal sender _
    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles Button4.Click
    FolderBrowserDialog1.SelectedPath = _
    Me.TextBox2.Text
    If FolderBrowserDialog1.ShowDialog() = _
    Windows.Forms.DialogResult.OK Then
        Me.TextBox2.Text = _
        FolderBrowserDialog1.SelectedPath
    End If
End Sub
```

6. modificate la prima istruzione del codice di gestione dell'evento Click di Button1 come segue:

```
OpenFileDialog1.InitialDirectory = _
Me.TextBox2.Text
```

7. modificate la prima istruzione del codice di gestione dell'evento Click di Button2 come segue:

```
SaveFileDialog1.InitialDirectory = _
Me.TextBox2.Text
```

8. avviate il programma con **F5**.

A questo punto abbiamo aggiunto anche un sistema per memorizzare la cartella corrente, per selezionare una cartella diversa e per crearne una nuova. Le modifiche apportate al codice permettono, inoltre, di leggere e salvare un file utilizzando il percorso selezionato, registrato nella casella di testo che abbiamo aggiunto con queste ultime modifiche.



La nostra applicazione per la registrazione delle ricette è praticamente completa di tutte le funzionalità essenziali per leggere, modificare, salvare, creare nuove ricette e perfino per selezionare l'attuale cartella di lavoro. Non resta che iniziare a inserire le ricette!

## FontDialog

Aggiungiamo ora una funzionalità in più alla nostra applicazione: vogliamo poter definire dinamicamente il font da utilizzare nelle caselle di testo. Infatti, il font predefinito è "Microsoft Sans Serif" di dimensione 8,25 punti: un po' troppo piccolo per una lettura agevole.

Ovviamente possiamo modificare le proprietà `Font.Name` e `Font.Size` delle caselle di testo, ed è quello che faremo con le seguenti modifiche:

1. aggiungete un controllo **FontDialog** (Figura 10.7);
2. inserite il seguente codice all'interno del codice del form:

```
Private Sub Form1_Load(ByVal sender _
    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles MyBase.Load
```

```
Dim f As New Font("Arial", "12")
Me.TextBox1.Font = f
Me.TextBox2.Font = f
End Sub
```

3. lanciate il programma con **F5**.

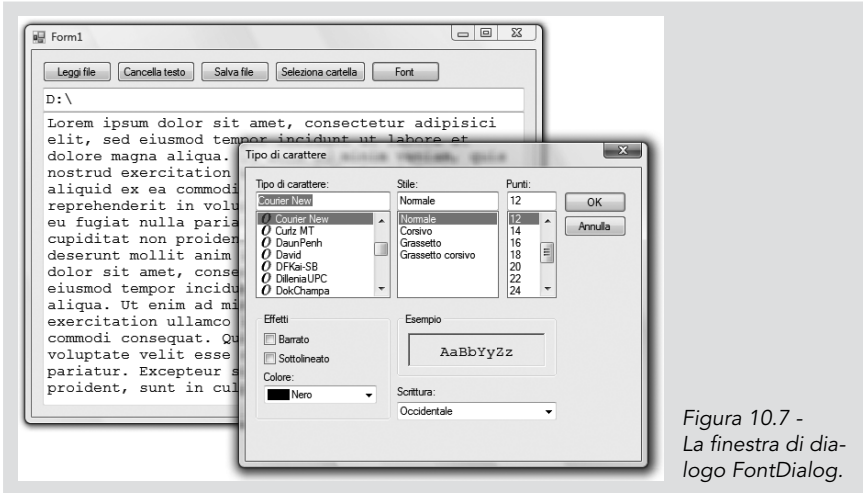


Figura 10.7 - La finestra di dialogo FontDialog.

Vedrete che i caratteri delle due caselle di testo sono ora più grandi e leggibili: l'evento Load del form viene scatenato al momento del caricamento del form, quindi il font delle due caselle di testo viene modificato prima della visualizzazione del form a schermo.

Vogliamo però avere la possibilità di modificare il font anche durante l'esecuzione del programma, quindi procediamo come segue:

1. aggiungete un pulsante (Button5) e modificate la sua proprietà Text in Font;
2. aggiungete un controllo **FontDialog**;
3. aggiungete il seguente codice nell'evento Click di Button5:

```
FontDialog1.ShowColor = True
FontDialog1.Font = TextBox1.Font
FontDialog1.Color = TextBox1.ForeColor

If FontDialog1.ShowDialog() <> _
    Windows.Forms.DialogResult.Cancel Then
```

```

TextBox1.Font = FontDialog1.Font
TextBox2.Font = FontDialog1.Font
TextBox1.ForeColor = FontDialog1.Color
TextBox2.ForeColor = FontDialog1.Color
End If

```

Come abbiamo visto, il controllo `FontDialog` permette di scegliere il tipo di font, la sua dimensione, il colore e altri particolari attributi, come il corsivo, il grassetto, il sottolineato e il barrato.

Naturalmente le caratteristiche del font scelte con questo controllo vengono applicate all'intera casella di testo. Quest'ultima, infatti, non permette di modificare le caratteristiche del testo selezionato.

## ColorDialog

Questo controllo ci permette di mostrare all'utente una finestra di dialogo per la scelta di un colore tra tutti quelli disponibili. È altresì possibile aprire la finestra di personalizzazione per selezionare un colore non compreso tra quelli predefiniti (Figura 10.8).

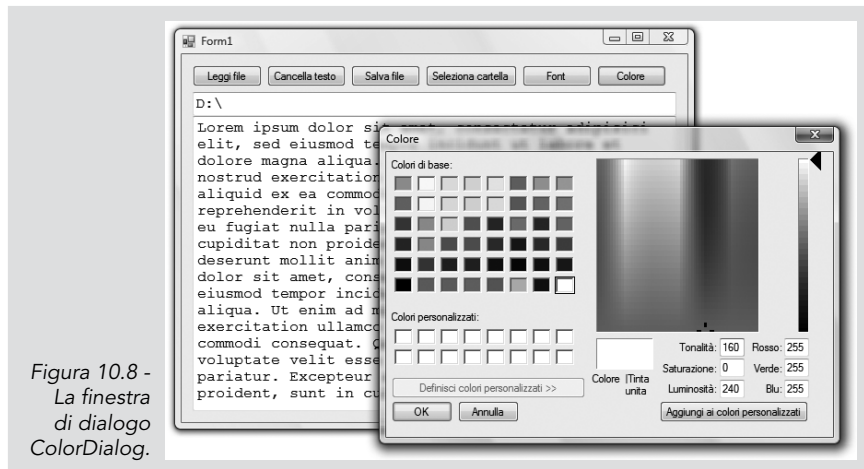


Figura 10.8 -  
La finestra  
di dialogo  
ColorDialog.

Modifichiamo ancora una volta il nostro editor, aggiungendo la possibilità di cambiare il colore del testo:

1. aggiungete il controllo **ColorDialog** al progetto;
2. aggiungete un pulsante (`Button6`) a `Form1`;

3. aggiungete il seguente codice per la gestione dell'evento Click del pulsante appena aggiunto:

```

` seleziona il colore iniziale uguale
` al colore di TextBox1:
ColorDialog1.Color = TextBox1.BackColor
` modifica il colore del testo di TextBox1
` con il colore selezionato:
If (ColorDialog1.ShowDialog() = _
    Windows.Forms.DialogResult.OK) Then
    TextBox1.ForeColor = ColorDialog1.Color
End If

```

4. avviate il programma premendo il tasto **F5**.

Potrete così constatare che è possibile assegnare al testo qualsiasi colore definito tra i colori di base, ma anche tra i colori personalizzati: è sufficiente fare clic sul pulsante **Definisci colori personalizzati >>** per visualizzare il controllo esteso con il pannello dei colori.

## Visto, si stampi!

La stampa di un documento è un'operazione indispensabile per molte applicazioni. Gli sviluppatori di Visual Basic e Visual Studio 2008 hanno quindi pensato bene di creare tre controlli specifici per eseguire in modo facile ed efficace questo compito.



Non è male l'idea di poter stampare le nostre ricette per poterle condividere con altri appassionati dell'arte culinaria!

Vediamo, quindi, quali sono le attività più comuni per la gestione della stampa (tra parentesi sono indicati i nomi dei controlli dedicati):

- ❖ impostazione della pagina (PageSetupDialog);
- ❖ anteprima di stampa (PrintPreviewDialog);
- ❖ stampa (PrintDialog).

Tutti questi controlli non sono inclusi nel gruppo **Finestre di Dialogo (Dialogs)**, come tutti i controlli visti finora, ma in un gruppo specifico della **Casella degli Strumenti (Toolbox)** denominato **Stampa (Printing)**. In questo gruppo sono presenti, come nella versione precedente di Visual Studio, anche i seguenti controlli:

- ❖ `PrintDocument`: per definire un oggetto che invia il proprio output alla stampante;
- ❖ `PrintPreviewControl`: rappresenta solo l'area di anteprima del documento, senza alcun pulsante né finestre di dialogo.

## PageSetupDialog

Indubbiamente, la possibilità di definire le varie impostazioni di pagina è necessaria per stampare qualsiasi documento.

Mediante il controllo `PageSetupDialog` è possibile definire il formato della pagina, l'alimentazione dei fogli, l'orientamento verticale/orizzontale e i margini (Figura 10.9).

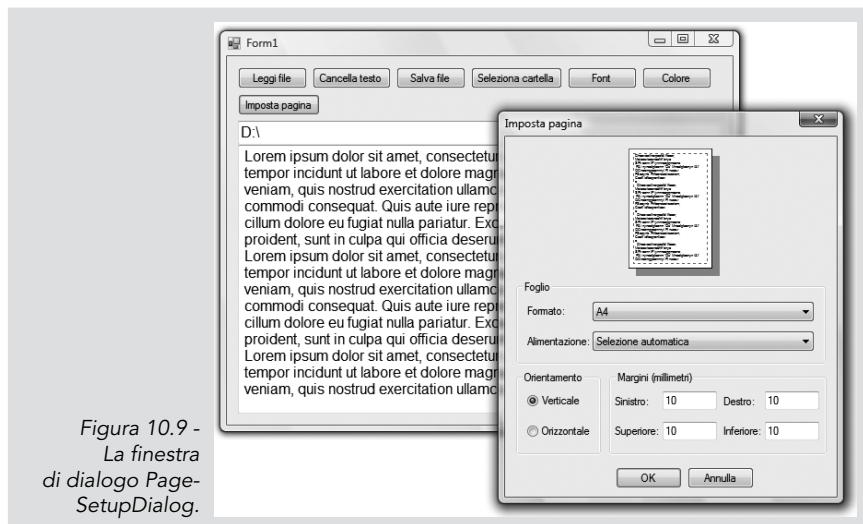


Figura 10.9 -  
La finestra  
di dialogo `PageSetupDialog`.

Vediamo come si può aggiungere questa possibilità al nostro editor:

1. aggiungete un controllo **PageSetupDialog** al progetto;

- aggiungete un pulsante (Button7) modificando il form come nella Figura 10.10;

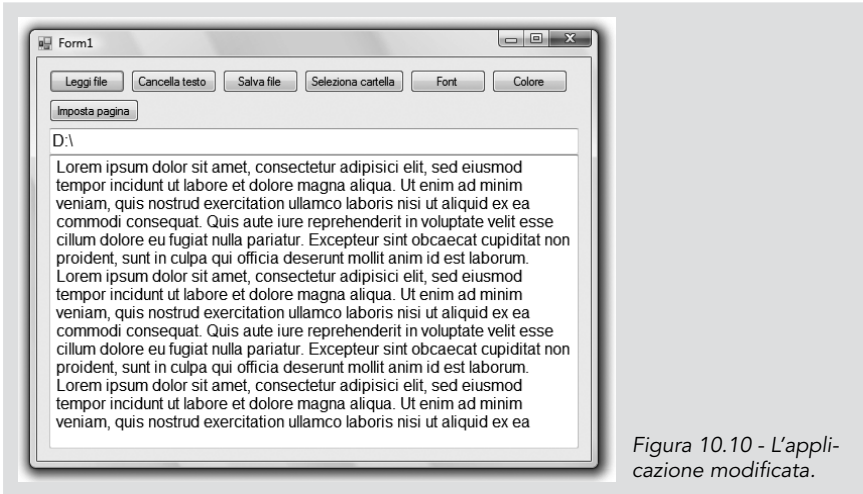


Figura 10.10 - L'applicazione modificata.

- aggiungete il seguente codice per la gestione dell'evento Click del pulsante appena aggiunto:

```

` inizializza il controllo con le impostazioni
` attuali della stampante predefinita
PageSetupDialog1.PageSettings = _
    New System.Drawing.Printing.PageSettings
PageSetupDialog1.PrinterSettings = New _
    System.Drawing.Printing.PrinterSettings
` apre la finestra di dialogo:
PageSetupDialog1.ShowDialog()
    
```

- premendo il tasto **F5** avvierete il programma.

Le impostazioni selezionate verranno così memorizzate all'interno del controllo PageSetupDialog e potranno essere lette o modificate utilizzando delle proprietà, tra le quali riportiamo quelle più utilizzate:

- ❖ PageSetupDialog1.PageSettings.Margins: margini della pagina;
- ❖ PageSetupDialog1.PageSettings.PaperSize: formato della pagina;

- ❖ `PageSetupDialog1.PageSettings.Landscape`: orientamento della pagina;
- ❖ `PageSetupDialog1.PrinterSettings.PrinterName`: nome della stampante;
- ❖ `PageSetupDialog1.PrinterSettings.PrintRange`: intervallo delle pagine da stampare.

## PrintPreviewDialog

È un controllo estremamente utile per realizzare un'anteprima di stampa di un documento creato da codice.

Questa funzionalità, ormai disponibile in quasi tutte le applicazioni in circolazione, è essenziale soprattutto per evitare di consumare carta inutilmente quando vogliamo fare delle prove di stampa prima della versione definitiva. In questo modo otteniamo una maggiore velocità di esecuzione, un risparmio di denaro e un impatto ecologico non trascurabile.



Prima di stampare le vostre ricette, controllate che la stampa sia corretta e completa. Le foreste vi ringrazieranno!

Per aggiungere questa funzionalità al nostro editor potete seguire questi semplici passi:

1. aggiungete un controllo **PrintPreviewDialog**;
2. aggiungete un pulsante (`Button8`) e assegnategli la proprietà `Text = "Anteprima"`;
3. aggiungete il seguente codice subito dopo la dichiarazione del form, per creare un oggetto che costituirà il documento da stampare:

```
Public WithEvents documento _
    As New System.Drawing.Printing.PrintDocument
```

4. aggiungete il seguente codice nella parte finale:

```
Private Sub Button8_Click(ByVal sender _
    As System.Object, _
```

```

        ByVal e As System.EventArgs) _
            Handles Button8.Click
    ` imposta la proprietà
    ` PrintPreviewDialog.Document all'oggetto
    ` PrintDocument selezionato dall'utente
    PrintPreviewDialog1.Document = documento
    ` chiama il metodo ShowDialog che scatterà
    ` l'evento PrintPage del documento
    PrintPreviewDialog1.ShowDialog()
End Sub

` inizializza il controllo
Private Sub InizializzaPrintPreviewDialog()
    ` imposta dimensione, posizione e nome
    Me.PrintPreviewDialog1.ClientSize = _
        New System.Drawing.Size(400, 300)
    Me.PrintPreviewDialog1.Location = _
        New System.Drawing.Point(29, 29)
    Me.PrintPreviewDialog1.Name = _
        "PrintPreviewDialog1"
    ` imposta la dimensione minima della finestra
    ` di dialogo
    Me.PrintPreviewDialog1.MinimumSize = _
        New System.Drawing.Size(375, 250)
    ` imposta la proprietà UseAntiAlias a True
    ` per permettere al sistema operativo di
    ` utilizzare l'effetto antialiasing
    Me.PrintPreviewDialog1.UseAntiAlias = True
End Sub

Private Sub document_PrintPage(ByVal sender _
    As Object, ByVal e As _
    System.Drawing.Printing.PrintPageEventArgs) _
    Handles documento.PrintPage
    ` Il seguente codice prepara un semplice
    ` messaggio sul documento creato nella
    ` finestra di dialogo
    Dim testo As String = Environment.NewLine & _
        "Sito dell'autore: www.deghetto.it"
    Dim printFont As New System.Drawing.Font _
        ("Arial", 25, _
        System.Drawing.FontStyle.Regular)
    e.Graphics.DrawString(testo, printFont, _
        System.Drawing.Brushes.Black, 0, 0)
    testo = Me.TextBox1.Text
    printFont = New System.Drawing.Font _
        ("Arial", 10, _
        System.Drawing.FontStyle.Regular)
    Dim rettangolo As _
        New RectangleF(0, 100, 800, 1100)
    e.Graphics.DrawString(testo, printFont, _

```

System.Drawing.Brushes.Black, rettangolo)  
 End Sub

Richiamiamo la vostra attenzione sulle ultime due istruzioni: la prima definisce un rettangolo entro il quale deve essere stampato il testo. I valori passati come parametri indicano rispettivamente la coordinata x, la coordinata y, la larghezza e l'altezza del rettangolo;

5. inserite il seguente codice nel gestore di evento Load del form:

```
documento = _
    New System.Drawing.Printing.PrintDocument
    InizializzaPrintPreviewDialog()
```

6. premete **F5** per eseguire il programma;

7. aprite un file di testo per visualizzarlo nella casella di testo e poi premere il pulsante di anteprima di stampa.

Sarà possibile vedere una finestra di anteprima del documento così creato (Figura 10.11). Tale finestra è altresì dotata di alcuni pulsanti per controllare la visualizzazione del documento (per esempio lo zoom) e la stampa.

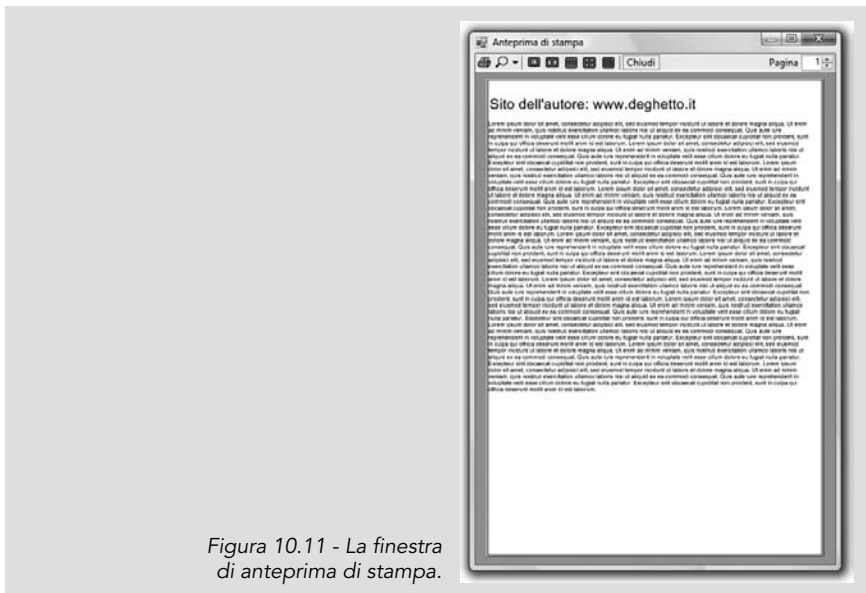


Figura 10.11 - La finestra di anteprima di stampa.

## PrintDialog

Dopo aver visto come gestire le impostazioni della stampante e della pagina e come visualizzare un'anteprima di stampa, non ci resta che stampare il nostro documento. In questo caso utilizziamo `PrintDialog`, un controllo che permette di scegliere una stampante di destinazione e di selezionare le parti del documento da stampare.

Estendiamo ancora una volta il nostro editor:

1. aggiungete un controllo **PrintDialog**;
2. aggiungete un pulsante (`Button9`) al form e assegnategli la proprietà `Text = "Stampa"`;
3. aggiungete il codice del gestore di evento `Click` del nuovo pulsante:

```

` permette all'utente di selezionare l'intervallo
` delle pagine da stampare
PrintDialog1.AllowSomePages = True
` imposta la proprietà Document a PrintDocument
PrintDialog1.Document = documento
` se l'utente preme il pulsante OK
` viene stampato il documento
If (PrintDialog1.ShowDialog() = _
    Windows.Forms.DialogResult.OK) Then
    documento.Print()
End If

```

4. premete **F5** per eseguire il programma;
5. aprite un file di testo e premete il pulsante di stampa;
6. selezionate la stampante, scegliete gli eventuali altri parametri desiderati e premete il pulsante **Stampa**.



Se avete installato una stampante virtuale PDF, in questo modo potrete creare in modo relativamente semplice dei file PDF da stampare, archiviare o inviare via e-mail.

## PrintDocument

Ricorderete senz'altro che nella prima parte del codice del form abbiamo aggiunto la seguente istruzione:

```
Public WithEvents documento _
    As New System.Drawing.Printing.PrintDocument
```

Questo codice è necessario per creare un oggetto di tipo `PrintDocument` che rappresenta il documento da creare e da stampare.

È possibile omettere tale istruzione aggiungendo semplicemente un controllo di tipo `PrintDocument` e assegnandogli la proprietà `Nome` (`Name`) desiderata: nel nostro caso sarà `documento`.

## PrintPreviewControl

L'ultimo controllo che esaminiamo, nell'ambito dei controlli disponibili per la gestione delle stampe, è `PrintPreviewControl`.

Esso permette di inserire un'area di anteprima di stampa nel form. Quest'area di anteprima sarà, però, priva di pulsanti e altri controlli.

`PrintPreviewControl` si differenzia quindi da `PrintPreviewDialog` perché lascia al programmatore l'onere di definire gli strumenti per controllare l'anteprima di stampa.

Per vedere un esempio di anteprima di stampa semplice, modifichiamo ancora una volta il nostro editor:

1. aggiungete un nuovo form lasciando il nome predefinito (`Form2`);
2. aggiungete a tale form un controllo **PrintPreviewControl** e modificate la proprietà `Dock` di questo controllo impostandola a `Fill`. Questa proprietà farà sì che il controllo riempi completamente l'area del form, anche nel caso in cui il form dovesse essere ridimensionato;
3. tornate sul `Form1` e aggiungete un nuovo pulsante (`Button10`), impostando la sua proprietà `Text` a `Anteprima` su `Form2`;
4. aggiungete il seguente codice al gestore dell'evento `Click` del nuovo pulsante:

```
' imposta la proprietà
' PrintPreviewDialog.Document all'oggetto
' PrintDocument selezionato dall'utente
Form2.PrintPreviewControl1.Document = documento
' chiama il metodo ShowDialog che scatterà
' l'evento PrintPage del documento
Form2.ShowDialog()
```

5. avviate il programma premendo il tasto **F5**;

6. aprite un file di testo, per visualizzarne il contenuto;
7. premete il pulsante Anteprema su Form2.

Potrete così veder apparire un form contenente solamente l'area di anteprema di stampa con il documento visualizzato.

## Paginazione

Finora abbiamo creato documenti formati solamente da un'unica pagina, ma in molti casi è necessario produrre documenti multipagina. Questo risultato è ottenibile in modo relativamente semplice utilizzando la proprietà `HasMorePages` di un oggetto di tipo `System.Drawing.Printing.PrintPageEventArgs` nel gestore dell'evento `PrintPage` del documento.

Proviamo quindi ad aggiungere questa funzionalità al nostro editor:

1. aggiungete la seguente istruzione all'inizio del codice di `Form1`:

```
Public pagina As Integer = 1
```

2. modificate il codice del metodo `document_PrintPage`, gestore dell'evento `PrintPage`, come segue:

```
Private Sub document_PrintPage(ByVal sender _
    As Object, ByVal e As _
    System.Drawing.Printing.PrintPageEventArgs) _
    Handles documento.PrintPage
    ' Il seguente codice prepara un semplice
    ' messaggio sul documento creato nella
    ' finestra di dialogo
    Dim testo As String = ""
    Dim printFont As _
        New System.Drawing.Font("Arial", 12)
    If pagina = 1 Then
        testo = Environment.NewLine & _
            "Sito dell'autore: www.deghetto.it"
        printFont = New System.Drawing.Font _
            ("Arial", 25, _
            System.Drawing.FontStyle.Regular)
        e.Graphics.DrawString(testo, printFont, _
            System.Drawing.Brushes.Black, 0, 0)
        pagina = 2
        e.HasMorePages = True
    ElseIf pagina = 2 Then
        testo = Me.TextBox1.Text
```

```

printFont = New System.Drawing.Font _
    ("Arial", 10, _
    System.Drawing.FontStyle.Regular)
Dim rettangolo As _
    New RectangleF(0, 100, 800, 1100)
e.Graphics.DrawString(testo, printFont, _
    System.Drawing.Brushes.Black, _
    rettangolo)
pagina = 1
e.HasMorePages = False
Else
    pagina = 1
    e.HasMorePages = False
End If
End Sub

```

3. avviate il programma premendo il tasto **F5**;
4. aprite un file di testo, per visualizzarne il contenuto nella casella di testo TextBox1;
5. premete il pulsante **Anteprima**.

Potrete così constatare che il documento è ora formato da due pagine: nella prima viene mostrato l'indirizzo dell'autore e nella seconda il testo preso dalla casella di testo TextBox1.

Dopo tutte le modifiche apportate, l'aspetto finale del nostro editor in modalità di progettazione è quello visibile nella Figura 10.12.



Figura 10.12 - L'editor in modalità progettazione.

## Un menu per tutti i gusti

Una selezione delle funzionalità desiderate basata solo su pulsanti può diventare scomoda se le funzionalità disponibili iniziano a essere molte, come nel caso nel nostro esempio di editor.

In questi casi è sempre opportuno organizzare le varie voci disponibili in un menu da posizionare nella parte alta del form.



Cosa c'è di meglio di avere un menu in un'applicazione che per gestire le nostre ricette di cucina? In questo caso, però, il menu serve per gestire l'applicazione, non per presentare i nostri piatti!

In questa ultima parte del capitolo, quindi, cercheremo di eliminare i pulsanti inseriti nel form per sostituirli con un più intuitivo e funzionale menu. Fino alla versione 2003 di Visual Studio e Visual Basic la gestione dei menu era piuttosto farraginoso e decisamente scomoda.

Poi finalmente, con la versione 2005, Microsoft ha reso disponibile un menu visuale estremamente agile e facile da gestire.

### MenuStrip

Il controllo `MenuStrip` permette di creare e modificare un menu in modo visuale, direttamente nel posto in cui si trova.

Per prima cosa è necessario aggiungere il controllo `MenuStrip` al form che vogliamo dotare di un menu.

Dopo questa operazione noteremo che nella parte alta del form è apparsa una striscia di colore celeste contenente una casella di testo bianca e, all'interno di questa, il testo "Digitare qui (Type here)".

Con un clic nella casella di testo notiamo che è possibile scrivere direttamente il testo appartenente alla voce di menu. Contemporaneamente appaiono altre due caselle di testo, una a destra e una sotto alla voce di menu, per permettere di inserire una nuova voce principale e una voce secondaria per il menu attuale (Figura 10.13).

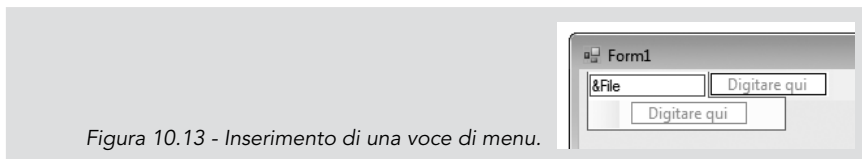


Figura 10.13 - Inserimento di una voce di menu.

Possiamo notare che la voce di menu include anche il simbolo &: esso indica al controllo `MenuStrip` che il carattere successivo è un tasto di scorciatoia, richiamabile in combinazione con il tasto **Alt**. Per esempio, nel caso della Figura 10.13, il menu **File** può essere richiamato anche da tastiera con la combinazione di tasti **Alt+F**.

È possibile definire anche quale tipo di controllo deve essere utilizzato per una voce di menu. Per selezionare il tipo di controllo da usare è possibile fare clic su una freccia che troviamo nel lato destro della voce di menu (Figura 10.14).

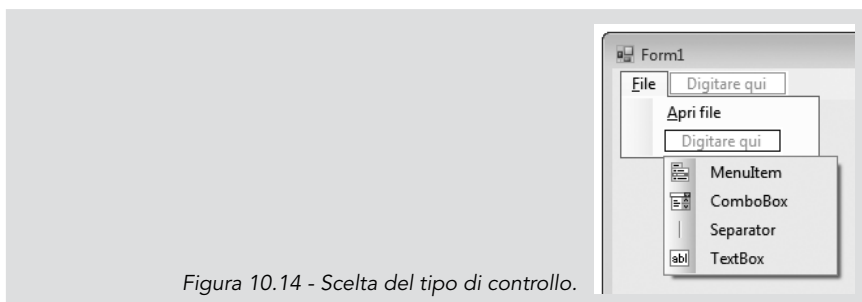


Figura 10.14 - Scelta del tipo di controllo.

I controlli disponibili sono:

- ❖ `MenuItem`: è una normale voce di menu, a sua volta espandibile con un sottomenu (Figura 10.15);

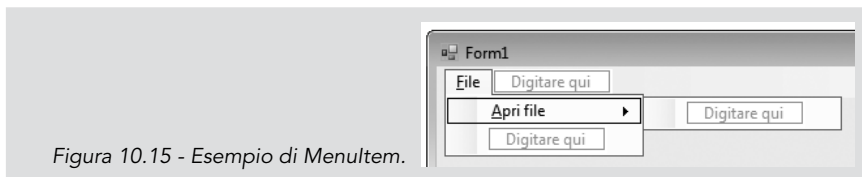


Figura 10.15 - Esempio di `MenuItem`.

- ❖ **ComboBox**: è la classica casella combinata, nella quale devono essere inserite le voci che vogliamo rendere disponibili all'utente. Per inserire tali voci, possiamo selezionare la proprietà `Items` e inserirle, ognuna in una riga diversa, nella casella di testo che apparirà premendo il pulsante con i tre punti (Figura 10.16);

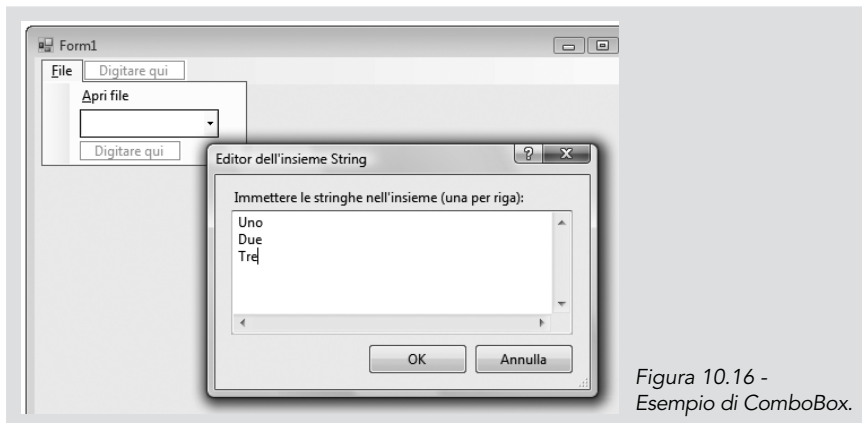


Figura 10.16 - Esempio di ComboBox.

- ❖ **Separator**: è una semplice linea orizzontale di separazione, per raggruppare nello stesso menu funzionalità di tipo diverso;
- ❖ **TextBox**: è una casella di testo che permette di inserire il testo desiderato, da utilizzare come riferimento nel form.

Iniziamo ora a modificare il nostro editor, per sostituire i vari pulsanti con le opportune voci di menu.

Dopo aver inserito le voci di menu principali (**&File**, **&Formato**, **&Stampa** e **&?**), iniziamo a inserire i controlli per ciascuna voce secondaria, come potete vedere dalla Figura 10.17.

Ora non resta che definire le azioni da eseguire quando l'utente farà clic sulle voci di menu. Per inserire il codice del gestore dell'evento `Click` di ogni voce di menu è sufficiente fare doppio clic con il mouse sulla voce corrispondente, così come faremmo con qualsiasi altro controllo posizionato sul form.

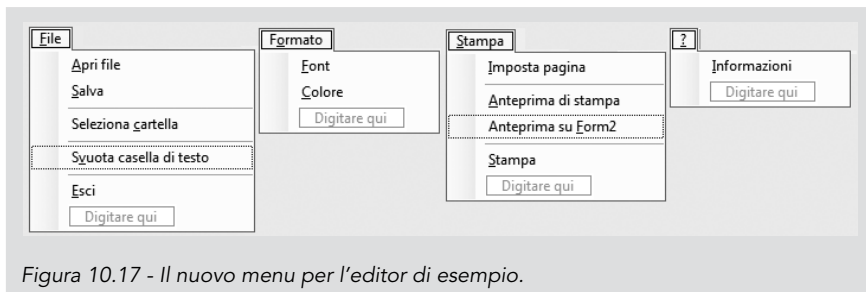


Figura 10.17 - Il nuovo menu per l'editor di esempio.

A questo punto dobbiamo solamente trasferire il codice di ciascun pulsante nelle corrispondenti voci di menu, eliminando progressivamente i pulsanti che non servono più.

Infine, aggiungiamo solamente il codice per la gestione delle voci di menu **Esci** e **Informazioni**.



Dato che l'applicazione è estremamente semplice e non presenta un utilizzo critico di risorse, per brevità utilizzeremo l'istruzione *End* per chiuderla. Ricordate però che in un'applicazione reale questo non dovrebbe mai essere fatto.

Il codice finale dell'applicazione è il seguente:

```
Public Class Form1
    Public pagina As Integer = 1

    Private Sub Form1_Load(ByVal sender _
        As System.Object, _
        ByVal e As System.EventArgs) _
        Handles MyBase.Load
        documento = New _
            System.Drawing.Printing.PrintDocument
        InizializzaPrintPreviewDialog()
        Dim f As New Font("Arial", "12")
        Me.TextBox1.Font = f
        Me.TextBox2.Font = f
    End Sub

    ' inzializza il controllo
    Private Sub InizializzaPrintPreviewDialog()
        ' imposta dimensione, posizione e nome
```

```

Me.PrintPreviewDialog1.ClientSize = _
    New System.Drawing.Size(400, 300)
Me.PrintPreviewDialog1.Location = _
    New System.Drawing.Point(29, 29)
Me.PrintPreviewDialog1.Name = _
    "PrintPreviewDialog1"
    ` imposta la dimensione minima della
    ` finestra di dialogo
Me.PrintPreviewDialog1.MinimumSize = _
    New System.Drawing.Size(375, 250)
    ` imposta la proprietà UseAntiAlias a
    ` True per permettere al sistema
    ` operativo di utilizzare l'effetto
    ` antialiasing
Me.PrintPreviewDialog1.UseAntiAlias = _
    True
End Sub

Private Sub document_PrintPage(ByVal _
    sender As Object, _
    ByVal e As _
System.Drawing.Printing.PrintPageEventArgs) _
    Handles documento.PrintPage
    ` Il seguente codice prepara un semplice
    ` messaggio sul documento creato nella
    ` finestra di dialogo
    Dim testo As String = ""
    Dim printFont As _
        New System.Drawing.Font("Arial", 12)
    If pagina = 1 Then
        testo = Environment.NewLine & _
            "Sito dell'autore: " & _
            "www.deghetto.it"
        printFont = New System.Drawing.Font _
            ("Arial", 25, _
            System.Drawing.FontStyle.Regular)
        e.Graphics.DrawString(testo, _
            printFont, _
            System.Drawing.Brushes.Black, _
            0, 0)
        pagina = 2
        e.HasMorePages = True
    ElseIf pagina = 2 Then
        testo = Me.TextBox1.Text
        printFont = New System.Drawing.Font _
            ("Arial", 10, _
            System.Drawing.FontStyle.Regular)
        Dim rettangolo As New RectangleF(0, _
            100, 800, 1100)
        e.Graphics.DrawString(testo, _
            printFont, _

```

```

        System.Drawing.Brushes.Black, _
        rettangolo)
    pagina = 1
    e.HasMorePages = False
Else
    pagina = 1
    e.HasMorePages = False
End If
End Sub

Private Sub ApriFileToolStripMenuItem_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ApriFileToolStripMenuItem.Click
    OpenFileDialog1.InitialDirectory = _
        Me.TextBox2.Text
    OpenFileDialog1.Filter = _
        "file di testo (*.txt)|*.txt|" & _
        "tutti i file (*.*)|*.*"
    OpenFileDialog1.FilterIndex = 1
    OpenFileDialog1.RestoreDirectory = True

    If OpenFileDialog1.ShowDialog() = _
        Windows.Forms.DialogResult.OK Then
        ` codice per leggere il contenuto del file
        Me.TextBox2.Text = OpenFileDialog1.FileName
        Me.TextBox1.Text = _
            My.Computer.FileSystem.ReadAllText("" & _
                OpenFileDialog1.FileName)
    End If
End Sub

Private Sub SvuotaCasellaDitestoToolStripMenuItem_Click(ByVal
sender _
    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles SvuotaCasellaDitestoToolStripMenuItem.Click
    Me.TextBox1.Text = ""
End Sub

Private Sub SalvaToolStripMenuItem_Click(ByVal _
    sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles SalvaToolStripMenuItem.Click
    SaveFileDialog1.InitialDirectory = _
        Me.TextBox2.Text
    SaveFileDialog1.Filter = _
        "file di testo (*.txt)|*.txt|" & _
        "tutti i file (*.*)|*.*"
    SaveFileDialog1.FilterIndex = 2
    SaveFileDialog1.RestoreDirectory = True

```

```

    If SaveFileDialog1.ShowDialog() = _
        Windows.Forms.DialogResult.OK Then
My.Computer.FileSystem.WriteAllText (" & _
        SaveFileDialog1.FileName, _
        Me.TextBox1.Text, False)
    End If
End Sub

Private Sub SelezionacartellaToolStripMenuItem_Click(ByVal
sender _
    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles SelezionacartellaToolStripMenuItem.Click
FolderBrowserDialog1.SelectedPath = _
    Me.TextBox2.Text
If FolderBrowserDialog1.ShowDialog() = _
    Windows.Forms.DialogResult.OK Then
    Me.TextBox2.Text = _
        FolderBrowserDialog1.SelectedPath
    End If
End Sub

Private Sub FontToolStripMenuItem_Click(ByVal sender As System.
Object, _
    ByVal e As System.EventArgs) _
    Handles FontToolStripMenuItem.Click
FontDialog1.ShowColor = True
FontDialog1.Font = TextBox1.Font
FontDialog1.Color = TextBox1.ForeColor

If FontDialog1.ShowDialog() <> _
    Windows.Forms.DialogResult.Cancel _
    Then
    TextBox1.Font = FontDialog1.Font
    TextBox2.Font = FontDialog1.Font
    TextBox1.ForeColor = FontDialog1.Color
    TextBox2.ForeColor = FontDialog1.Color
    End If
End Sub

Private Sub ColorToolStripMenuItem_Click(ByVal sender _
    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ColorToolStripMenuItem.Click
' seleziona il colore iniziale uguale al
' colore di TextBox1:
ColorDialog1.Color = TextBox1.BackColor
' modifica il colore del testo di TextBox1
' con il colore selezionato:
If (ColorDialog1.ShowDialog() = _
    Windows.Forms.DialogResult.OK) Then

```

```

        TextBox1.ForeColor = ColorDialog1.Color
    End If
End Sub

Private Sub ImpostaPaginaToolStripMenuItem_Click(ByVal sender _
    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles _
    ImpostaPaginaToolStripMenuItem.Click
    ` inizializza il controllo con le
    ` impostazioni attuali della stampante
    ` predefinita
    PageSetupDialog1.PageSettings = _
        New System.Drawing.Printing.PageSettings
    PageSetupDialog1.PrinterSettings = New _
        System.Drawing.Printing.PrinterSettings
    ` apre la finestra di dialogo
    PageSetupDialog1.ShowDialog()
End Sub

Private Sub AntepremaDiStampaToolStripMenuItem_Click(ByVal
sender _
    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles _
    AntepremaDiStampaToolStripMenuItem.Click
    ` imposta la proprietà
    ` PrintPreviewDialog.Document all'oggetto
    ` PrintDocument selezionato dall'utente
    PrintPreviewDialog1.Document = documento
    ` chiama il metodo ShowDialog che scatenerà
    ` l'evento PrintPage del documento
    PrintPreviewDialog1.ShowDialog()
End Sub

Private Sub AntepremaSuForm2ToolStripMenuItem_Click(ByVal _
    sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles _
    AntepremaSuForm2ToolStripMenuItem.Click
    ` imposta la proprietà
    ` PrintPreviewDialog.Document all'oggetto
    ` PrintDocument selezionato dall'utente
    Form2.PrintPreviewControl1.Document = _
        documento
    ` chiama il metodo ShowDialog che scatenerà
    ` l'evento PrintPage del documento
    Form2.Show()
End Sub

Private Sub StampaToolStripMenuItem_Click(ByVal sender _

```

```

    As System.Object, _
    ByVal e As System.EventArgs) _
    Handles StampaToolStripMenuItem.Click
` permette all'utente di selezionare
` l'intervallo delle pagine da stampare
PrintDialog1.AllowSomePages = True

` imposta la proprietà Document a
` PrintDocument
PrintDialog1.Document = documento

` se l'utente preme il pulsante OK
` viene stampato il documento
If (PrintDialog1.ShowDialog() = _
    Windows.Forms.DialogResult.OK) Then
    documento.Print()
End If
End Sub

Private Sub InformazioniToolStripMenuItem_Click(ByVal _
    sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles _
    InformazioniToolStripMenuItem.Click
    MessageBox.Show("Editor di file " & _
        "di testo " & Environment.NewLine & _
        "by Mario De Ghetto (www.deghetto.it)")
End Sub
End Class

```

Notate come è composto il nome dei gestori degli eventi `Click` delle singole voci di menu: una prima parte ripete il nome che è stato dato alla voce di menu, quindi subito di seguito viene sempre riportato `ToolStripMenuItem.Click`. Un'eccezione è data dalla scelta di un controllo di tipo `MenuItem` dalla casella combinata: in tal caso, infatti, non essendo disponibile a priori un nome per la voce di menu, viene utilizzato un nome predefinito composto da `ToolStripMenuItem` seguito da un numero progressivo di inserimento. Ecco il motivo per il quale nel codice sopra riportato abbiamo, per esempio, un riferimento a un controllo `ToolStripMenuItem5`.

## StatusStrip

Un controllo `StatusStrip` permette di inserire una barra di stato nella parte inferiore del form.

Sostanzialmente funziona allo stesso modo di un controllo `MenuStrip`, tranne per i controlli disponibili che sono i seguenti:

- ❖ `StatusLabel`: è un'etichetta personalizzabile e permette di visualizzare informazioni utili all'utente;
- ❖ `ProgressBar`: è una barra di progressione per una certa attività e serve per indicare all'utente a che punto è arrivata l'elaborazione, nel caso in cui quest'ultima sia particolarmente dispendiosa in merito al tempo necessario per il suo completamento;
- ❖ `DropDownButton`: permette l'inserimento di una casella combinata definita con un elenco di voci, inserite solitamente in fase di progettazione;
- ❖ `SplitButton`: permette di eseguire un'azione predefinita oppure di selezionare una delle azioni disponibili nella casella combinata.

Aggiungiamo al nostro editor anche un controllo `StatusStrip`, poi inseriamo un controllo del tipo `StatusLabel` per rimuovere la casella di testo `TextBox2`. Dopo questa operazione, è sufficiente sostituire nel codice tutte le occorrenze di `TextBox2` con `ToolStripStatusLabel1` ed eliminare la casella di testo, ormai superflua.

In questo modo abbiamo eliminato anche la casella di testo, così antiestetica, e abbiamo inserito al suo posto un riferimento alla barra di stato che, in ogni momento, ci darà le informazioni sulla cartella e sul file attuale.

Nella Figura 10.18 potete vedere il risultato finale e apprezzare la possibilità di modificare anche il font e il colore delle etichette della barra di stato.

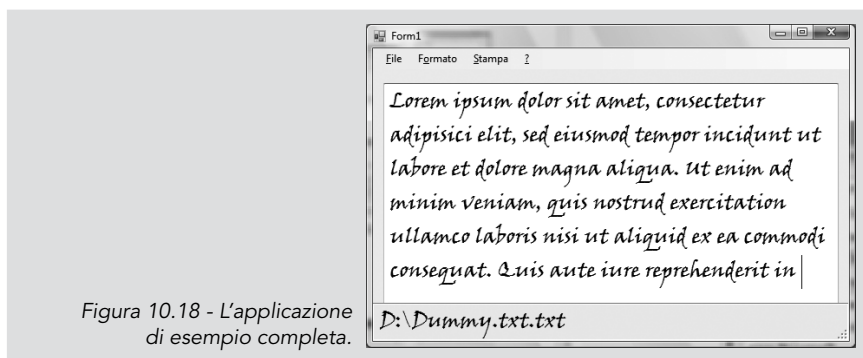


Figura 10.18 - L'applicazione di esempio completa.

ToolStripProgressBar è una valida alternativa al controllo ProgressBar, mantenuto solo per compatibilità. Consente di visualizzare una barra di progressione direttamente in un controllo StatusStrip, come nell'esempio seguente:

1. create un nuovo progetto;
2. inserite un pulsante e un controllo StatusStrip;
3. nel controllo StatusStrip selezionate un controllo di tipo ProgressBar. Il controllo così inserito si chiamerà, di default, ToolStripProgressBar1;
4. selezionate tale controllo e impostate la proprietà Visible = False;
5. nel codice dell'evento Click del pulsante, inserite il seguente codice:

```
Private Sub Button1_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles Button1.Click
    ToolStripProgressBar1.Visible = True
    For i As Integer = 1 To 100
        System.Threading.Thread.Sleep(100)
        ToolStripProgressBar1.Value = i
        Application.DoEvents()
    Next
    MessageBox.Show("Operazione terminata")
End Sub
```

6. eseguite il programma premendo il tasto **F5** e premete il pulsante che avete appena aggiunto.

Con l'esempio presentato vedrete la barra di progressione apparire e, lentamente, avanzare fino al completamento. Al termine apparirà un messaggio di conferma.

## Conclusioni

Questo capitolo ha presentato un'applicazione semplice ma completa di tutte le funzioni indispensabili per la gestione di file di testo. Naturalmente l'obiettivo di costruire un editor di testo è stato un pretesto per presentare molti nuovi controlli, utilissimi in numerose applicazioni.