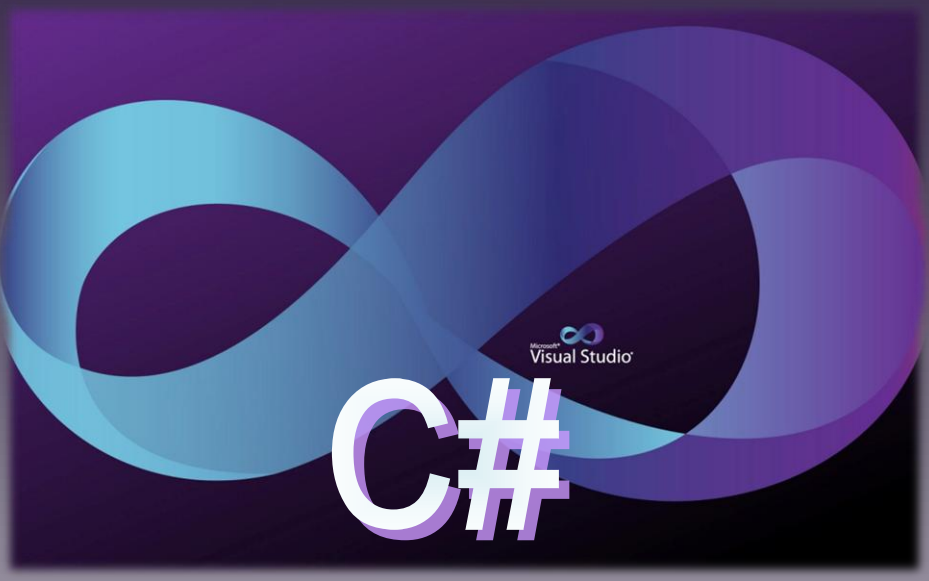
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

FILE, COMPONENTI E CONTROLLI

DISPENSA 12.02

12-02_File_e_Controlli_[15]

Versione del: 07/11/2015
Revisione numero: 15

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

CONTROLLI CHE USANO FILE

IL CONTROLLO RICHTEXTBOX

SCOPO E USO DEL CONTROLLO

DESCRIZIONE SINTETICA DEL CONTROLLO RICHTEXTBOX

Un controllo **RichTextBox** è una casella di testo avanzato che consente l'editing del testo e funzionalità di formattazione avanzate quali il formato di testo ricco di carico (RTF) i file.

In questa dispensa, è illustrato come creare il controllo e utilizzarne le varie funzionalità del controllo RichTextBox.

CREAZIONE DI UN CONTROLLO RICHTEXTBOX

È possibile creare un controllo RichTextBox sia utilizzando il designer Forms in fase di progettazione sia utilizzando la classe RichTextBox nel codice a run-time (fase di esecuzione).

Per creare un controllo RichTextBox in fase di progettazione, è sufficiente trascinare e rilasciare un controllo RichTextBox dalla Casella degli strumenti in un Form di Visual Studio. Una volta che un RichTextBox viene aggiunto a un Form, è possibile spostarlo e ridimensionarlo usualmente usando il mouse o impostando le sue proprietà.

La creazione di un controllo RichTextBox in fase di esecuzione corrisponde essenzialmente a creare un'istanza della classe RichTextBox, impostare le sue proprietà e aggiungerlo alla collezione Controls del Form. Il primo passo per creare un RichTextBox a tempo di esecuzione è quello di creare un'istanza della classe RichTextBox. Il frammento di codice seguente propone un esempio di creazione di un oggetto RichTextBox.

VC#

```
// Crea un oggetto RichTextBox
RichTextBox dynamicRichTextBox = nuovo RichTextBox ();
```

Nel passaggio successivo, si mostra come sia possibile impostare le proprietà di un controllo RichTextBox. Il frammento di codice seguente imposta dimensione, posizione, colore di sfondo, colore di primo piano, Testo, Nome, Font e le proprietà di un oggetto RichTextBox.

VC#

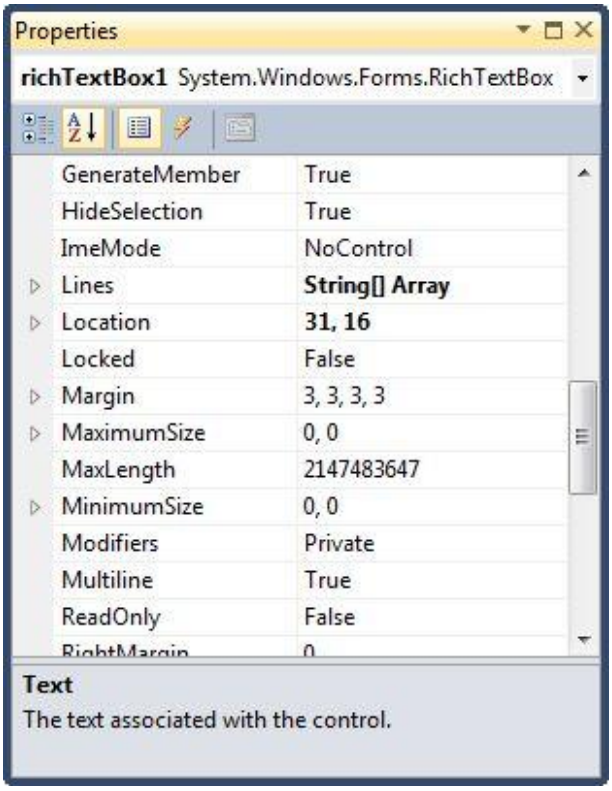
```
dynamicRichTextBox.Name = "DynamicRichTextBox" ;
dynamicRichTextBox.Location = new Point (20, 20);
dynamicRichTextBox.Width = 300;
dynamicRichTextBox.Height = 200;
// Imposto sfondo e primo piano
dynamicRichTextBox.BackColor = Color.Red;
dynamicRichTextBox.ForeColor = Color.Blue;
dynamicRichTextBox.Text = "Creato un RichTextBox al volo" ;
dynamicRichTextBox.Font = nuovo Font ( "Georgia" , 16);
```

Una volta che un controllo RichTextBox è pronto con le sue proprietà, il passo successivo è quello di agganciarlo al Form. Per farlo, si usa il metodo **Form.Controls.Add**. Il frammento di codice seguente aggiunge il controllo creato prima al Form corrente.

VC#
```
Controls.Add (dynamicRichTextBox);
```

L'aspetto del controllo RichTextBox è simile a quello della figura qui proposta. Ovviamente è possibile impostare le proprietà del controllo anche dopo averlo inserito nel Form.

PROPRIETÀ DEL RICHTEXTBOX



Il modo più immediato per impostare le proprietà a tempo di progettazione è utilizzare la finestra delle Proprietà.
È possibile aprire la finestra delle proprietà premendo il tasto F4 o fare clic destro su un controllo e selezionare il menu Proprietà. La finestra Proprietà assomiglia alla Figura qui di lato.

POSIZIONE DEL RICHTEXTBOX

La proprietà Location prende un punto che specifica la posizione di partenza del RichTextBox in un Form. Invece della proprietà Location è possibile usare le proprietà Top e Left.

VC#
```
dynamicRichTextBox.Location = nuovo punto (20, 20);
```

ALTEZZA, LARGHEZZA E DIMENSIONE DEL RICHTEXTBOX

La proprietà Size specifica la dimensione del controllo. Possiamo anche utilizzare la proprietà Width e Height, invece di Size. Il frammento di codice seguente imposta Width e Height del controllo RichTextBox.

VC#
```
dynamicRichTextBox.Width = 300;
dynamicRichTextBox.Height = 200;
```

BACKGROUND, FOREGROUND, BORDERSTYLE

BackColor e ForeColor si usano rispettivamente per impostare il colore di sfondo e di primo piano del RichTextBox. Se si fa clic su queste proprietà nella finestra Proprietà, si apre la finestra di dialogo colore.

In alternativa, è possibile impostare colori di sfondo e primo piano in fase di esecuzione, come nel seguente esempio:

VC#
```
// Set sfondo e primo piano
dynamicRichTextBox.BackColor = Color.Red;
dynamicRichTextBox.ForeColor = Color.Blue;
```

È anche possibile impostare lo stile dei bordi di un RichTextBox utilizzando la proprietà BorderStyle. La proprietà BorderStyle assume i valori elencati nella enumerazione di BorderStyle che ha tre valori: FixedSingle, Fixed3D, e None. Il valore predefinito di stile del bordo è Fixed3D.

Il frammento di codice seguente imposta lo stile del bordo di un RichTextBox su FixedSingle.

VC#
```
dynamicRichTextBox.BorderStyle = BorderStyle.FixedSingle;
```

PROPRIETÀ NAME DEL RICHTEXTBOX

La proprietà Name rappresenta un nome univoco di un controllo RichTextBox. Esso è utilizzato per accedere al controllo del codice. Il frammento di codice seguente imposta e prende il nome e il testo di un controllo RichTextBox.

VC#
```
dynamicRichTextBox.Name = "DynamicRichTextBox" ;
```

PROPRIETÀ TESTO E TEXTLENGTH

La proprietà Text di un RichTextBox rappresenta il testo corrente di un controllo RichTextBox. La proprietà di sola lettura textLength restituisce la lunghezza di un contenuto di RichTextBox. Il frammento di codice seguente imposta le proprietà Text e TextAlign e ottiene la dimensione di un controllo RichTextBox.

VC#
```
dynamicRichTextBox.Text = "tEsTo vIsUaLiZzAtO nElLa CaSeLlA" ;
int size = dynamicRichTextBox.TextLength;
```

PROPRIETÀ ACCEPTSTAB

La proprietà AcceptsTab è booleana; se vale True il controllo accetta i caratteri che rappresentano le tabulazioni. Per impostazione predefinita, il valore della proprietà AcceptsTab di un controllo RichTextBox è False.

Se un controllo RichTextBox è predisposto per più righe, la proprietà AcceptsTab viene utilizzata per impostare il controllo RichTextBox di accettare il tasto TAB come testo. Se questa proprietà non è impostata, premendo il tasto TAB si muove semplicemente al controllo successivo sul modulo.

VC#
```
// Accetta il tasto TAB
dynamicRichTextBox.AcceptsTab = true ;
```

PROPRIETÀ SCROLLBARS

Un controllo RichTextBox Multiline può avere barre di scorrimento. La proprietà ScrollBars del controllo RichTextBox è utilizzata per stabilire quali barre di scorrimento mostrare per il controllo. La proprietà ScrollBars ammette i valori elencati nella enumerazione RichTextBoxScrollBars che propone quattro valori : Both, Vertical, Horizontal, and None.

Il frammento di codice seguente rende entrambe le barre di scorrimento verticali e orizzontali attivo su un controllo RichTextBox e saranno visibili quando è necessario lo scorrimento su un controllo RichTextBox.

VC#
```
// Entrambe le barre di scorrimento, sempre
dynamicRichTextBox.ScrollBars = RichTextBoxScrollBars.Both;
```

PROPRIETÀ WORDWRAP

La proprietà WordWrap è booleana. Se la proprietà WordWrap è true, il testo nel controllo RichTextBox passa automaticamente alla riga successiva, se abbastanza lungo. Se questa proprietà è impostata su true, le barre di scorrimento orizzontali non saranno visualizzate indipendentemente dall'impostazione della proprietà ScrollBars.

VC#
```
// A capo automatico
dynamicRichTextBox.WordWrap = true ;
```

PROPRIETÀ FONT DI UN CONTROLLO RICHTEXTBOX

La proprietà Font rappresenta il carattere del testo di un controllo RichTextBox. Se si fa clic sulla proprietà Font nella finestra Proprietà, compare il nome font, le dimensioni e le altre opzioni dei caratteri. Il frammento di codice seguente imposta la proprietà font a run-time.

VC#
```
// Imposta un nuovo Font
dynamicRichTextBox.Font = new Font ( "Georgia" , 16);
```

LUNGHEZZA MASSIMA DI TESTO

È possibile limitare il numero di caratteri ammessi in un controllo RichTextBox impostando la proprietà MaxLength. Il frammento di codice seguente imposta la lunghezza massima di un RichTextBox a 50 caratteri.

VC#
```
// Imposta la lunghezza massima del testo a 50 caratteri.
dynamicRichTextBox.MaxLength = 50;
```

PROPRIETÀ READONLY

È possibile effettuare un controllo RichTextBox di sola lettura (non modificabile) impostando la proprietà ReadOnly su true. Il frammento di codice seguente imposta la proprietà ReadOnly su true.

VC#
```
// Imposta un testo non modificsbile dall'utente
dynamicRichTextBox.ReadOnly = true ;
```

ATTIVAZIONE E DISATTIVAZIONE DI TASTI SCORCIATOIA

Un controllo RichTextBox consente l'uso di tasti scorciatoia. I tasti scorciatoia sono combinazioni da tastiera che causano azioni sul testo.

ShortcutsEnabled proprietà del RichTextBox viene utilizzato per abilitare o disabilitare le scorciatoie. Per impostazione predefinita, i collegamenti sono abilitati. Il frammento di codice seguente disattiva scorciatoie in una RichTextBox.

VC#
```
dynamicRichTextBox.ShortcutsEnabled = false ;
```

La proprietà **ShortcutsEnabled** si applica alle seguenti combinazioni di tasti di scelta rapida:

CTRL + A	
CTRL + BACKSPACE	Cancella paragrafo a sinistra
CTRL + C	Copia testo selezionato
CTRL + CANC	Cancella paragrafo a destra
CTRL + E	
CTRL + L	
CTRL + R	
CTRL + V	Incolla
CTRL + X	Taglia testo selezionato
CTRL + Y	Ripeti ultima azione
CTRL + Z	Annulla ultima azione
MAIUSC + CANC	
SHIFT + INSERT	

UTILIZZO DEL RICHTEXTBOX

AGGIUNTA DI TESTO

Un modo per aggiungere testo a un RichTextBox è semplicemente impostare la proprietà Text al testo corrente e il nuovo testo che si vorrebbe aggiungere qualcosa di simile.

VC#
```
// Aggiunge del testo alla fine
RichTextBox1.Text += "testo aggiunto" ;
```

RichTextBox dispone comunque anche del metodo AppendText che ha lo stesso effetto. Lo *AppendText* aggiunge testo metodo alla fine di un RichTextBox. Il frammento di codice seguente viene utilizzato il metodo AppendText per aggiungere testo alle RichTextBox1 contenuti.

VC#
```
// Aggiunge del testo alla fine
RichTextBox1.AppendText ( "testo aggiunto" );
```

CONTENUTO DEL RICHTEXTBOX

Il modo più semplice per leggere il contenuto del controllo RichTextBox è usare la proprietà Text. Si deve osservare, tuttavia, che la proprietà Text non ha alcuna formattazione, ma contiene solo testo. Per ottenere il testo con formattazione conviene sfruttare la proprietà Rtf. Il frammento di codice seguente copia il contenuto testuale di un RichTextBox in una variabile di tipo stringa.

VC#
```
string frase = RichTextBoxContents dynamicRichTextBox.Text;
```

In un RichTextBox righe, se il contenuto RichTextBox sono separati da linee multiple e volete leggere il contenuto di un RichTextBox linea per linea, è possibile utilizzare la proprietà Lines del RichTextBox. La proprietà Lines restituisce un array di stringhe in cui ciascun elemento della matrice restituita è una linea.

PROPRIETÀ LINES DEL RICHTEXTBOX

Il seguente frammento di codice legge una riga per riga il contenuto RichTextBox.

VC#
```
string [] miefrasi = dynamicRichTextBox.Lines;
foreach (string mialinea in miefrasi)
{
    MessageBox.Show(mialinea);
}
```

SELEZIONE IN RICHTEXTBOX

La proprietà SelectedText restituisce il testo selezionato in un controllo RichTextBox.

VC#
```
// copia il testo selezionato in una variabile
string mioTestoSelezionato = dynamicRichTextBox.SelectedText;
```

È inoltre possibile utilizzare le proprietà SelectionStart e SelectionLength per ottenere e impostare il testo selezionato in un controllo RichTextBox. La proprietà SelectionStart rappresenta l'indice iniziale del testo selezionato e SelectionLength proprietà rappresenta il numero di caratteri da selezionare dopo il carattere di partenza. Il frammento di codice seguente imposta la selezione su un RichTextBox.

VC#
```
// definisce il testo selezionato da programma
dynamicRichTextBox.SelectionStart = 10;
dynamicRichTextBox.SelectionLength = 20;
```

METODI CLEAR, SELECTALL E DESELECTALL

Il metodo Clear elimina il contenuto di un RichTextBox. Il frammento di codice seguente viene utilizzato il metodo Clear per cancellare il contenuto di un RichTextBox.

VC#
```
// pulisce il testo del RichTextBox1
RichTextBox1.Clear ();
```

RichTextBox classe fornisce i metodi e selectAll DeselectAll per selezionare e deselezionare tutto il testo di un controllo RichTextBox. Il frammento di codice seguente mostra come utilizzare i metodi e selectAll DeselectAll.

VC#
```
private void selectAllToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (RichTextBox1.TextLength > 0)
        RichTextBox1.SelectAll();
}
```

VC#
```
private void deselectAllToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (RichTextBox1.TextLength > 0)
        RichTextBox1.DeselectAll();
}
```

TAGLIA, COPIA, INCOLLA, ANNULLA OPERAZIONI IN RICHTEXTBOX

La classe RichTextBox fornisce diversi metodi di operazione sul testo:

Cut	Taglia	Sposta il testo selezionato nella clipboard
Copy	Copia	Copia il testo selezionato nella clipboard
Paste	Incolla	Incolla dalla clipboard al posto del testo selezionato
Undo	Annulla	Annulla l'ultima azione

Il frammento di codice seguente mostra come utilizzare i metodi.

VC#
```
private void cutToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (RichTextBox1.SelectionLength > 0)
        RichTextBox1.Cut();
}
```

VC#
```
private void copyToolStripMenuItem_Click(object sender, EventArgs e)
{
        if (RichTextBox1.SelectionLength > 0)
            RichTextBox1.Copy();
}
```

VC#
```
private void copyToolStripMenuItem_Click(object sender, EventArgs e)
{
        if (RichTextBox1.SelectionLength > 0)
            RichTextBox1.Copy();
}
```

VC#
```
private void undoToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (RichTextBox1.CanUndo)
    {
        RichTextBox1.Undo();
        RichTextBox1.ClearUndo();
    }
}
```

RIPETIZIONI DI OPERAZIONI SUL RICHTEXTBOX

Il metodo Ripeti può essere utilizzato per ripristinare l'ultima operazione di annullamento al controllo. La proprietà CanRedo informa se ci sono azioni che si sono eseguite all'interno del RichTextBox che possano essere riapplicate.

VC#

```
if ( dynamicRichTextBox.CanRedo == true )
{
    if ( dynamicRichTextBox.RedoActionName != "Elimina" )
        dynamicRichTextBox.Redo ();
}
```

PROPRIETÀ DI SELEZIONE DEL RICHTEXTBOX

Il frammento di codice seguente imposta le proprietà di selezione.

VC#

```
private void SelectionButton_Click(object sender, EventArgs e)
{
  dynamicRichTextBox.BackColor = Color.White;
  dynamicRichTextBox.Clear();
  dynamicRichTextBox.BulletIndent = 10;
  dynamicRichTextBox.SelectionFont = new Font("Georgia",16, FontStyle.Bold);
  dynamicRichTextBox.SelectedText = "Mindcracker Network \n";
  dynamicRichTextBox.SelectionFont = new Font("Verdana", 12);
  dynamicRichTextBox.SelectionBullet = true;
  dynamicRichTextBox.SelectionColor = Color.DarkBlue;
  dynamicRichTextBox.SelectedText = "C# Corner" + "\n";
  dynamicRichTextBox.SelectionFont = new Font("Verdana", 12);
  dynamicRichTextBox.SelectionColor = Color.Orange;
  dynamicRichTextBox.SelectedText = "VB.NET Heaven" + "\n";
  dynamicRichTextBox.SelectionFont = new Font("Verdana", 12);
  dynamicRichTextBox.SelectionColor = Color.Green;
  dynamicRichTextBox.SelectedText = ".Longhorn Corner" + "\n";
  dynamicRichTextBox.SelectionColor = Color.Red;
  dynamicRichTextBox.SelectedText = ".NET Heaven" + "\n";
  dynamicRichTextBox.SelectionBullet = false;
  dynamicRichTextBox.SelectionFont = new Font("Tahoma", 10);
  dynamicRichTextBox.SelectionColor = Color.Black;
  dynamicRichTextBox.SelectedText = "Testo slezionato.\n";
}
```

CARICARE E SALVARE FILE RTF

Il metodo **`LoadFile`** del controllo RichTextBox è utilizzato per caricare un file RTF e visualizzarne il contenuto al suo interno.

Il metodo **`SaveFile`** dello stesso controllo, serve per salvare il contenuto di un RichTextBox in un file RTF. Il frammento di codice seguente viene caricato un file RTF usando un OpenFileDialog e salva di nuovo il suo contenuto.

VC#
```
private void LoadRTFButton_Click(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.InitialDirectory = "c:\\";
    ofd.Filter = "txt files (*.txt)|*.txt|Tutti i files (*.*)|*.*";
    ofd.FilterIndex = 2;
    ofd.RestoreDirectory = true;
    if (ofd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        dynamicRichTextBox.LoadFile(ofd.FileName);
        dynamicRichTextBox.Find("Text", RichTextBoxFinds.MatchCase);
        dynamicRichTextBox.SelectionFont = new Font("Verdana", 12,
                                                    FontStyle.Bold);
        dynamicRichTextBox.SelectionColor = Color.Red;
        dynamicRichTextBox.SaveFile(@"C:\. . .\mioDocumento.rtf",

     RichTextBoxStreamType.RichText);
    }
}
```

LA PROPRIETÀ BULLETINDENT

La proprietà **BulletIndent** indica o imposta il rientro utilizzato nel controllo RichTextBox se viene applicato al testo lo stile di elenco puntato.

VC#
```
dynamicRichTextBox.BulletIndent = 10;
```

LA PROPRIETÀ DETECTURLS

Se è impostato su true, la proprietà **DetectUrls** serve per formattare automaticamente un Uniform Resource Locator (URL, cioè un indirizzo web) quando è digitato nel controllo.

LA PROPRIETÀ ENABLEAUTODRAGDROP

Il controllo RichTextBox supporta anche le operazioni drag and drop che ci permettono di trascinare e rilasciare testo, immagini e altri dati. La proprietà **EnableAutoDragDrop** consente di usare il drag-and-drop sul testo, su immagini e su altri dati.

VC#
```
dynamicRichTextBox.EnableAutoDragDrop = true ;
```

LE PROPRIETÀ RIGHTMARGIN, AUTOWORDSELECTION E ZOOMFACTOR

La proprietà **RightMargin** rappresenta la dimensione di una singola riga di testo all'interno di un controllo RichTextBox.

La proprietà **AutoWordSelection** indica se una parola è selezionata automaticamente quando dentro il testo si esegue un doppio click all'interno di un controllo RichTextBox.

La proprietà **ZoomFactor** indica il livello di zoom corrente del RichTextBox, ovvero il suo ingrandimento visivo. Il valore 1,0 significa che non c'è alcuno zoom applicato su un controllo.

VC#

```
private vuoto ZoomButton_Click ( object sender, EventArgs e)
{
    dynamicRichTextBox.AutoWordSelection = true ;
    dynamicRichTextBox.RightMargin = 5;
    dynamicRichTextBox.ZoomFactor = 3.0f;
}
```

LE PROPRIETÀ RTF E SELECTEDRTF

La proprietà Rtf del controllo permette di ottenere e impostare un testo formattato in stile Rich Text Format (RTF). Il testo RTF è il testo che include la formattazione.

RIEPILOGO SUL CONTROLLO RICHTEXTBOX

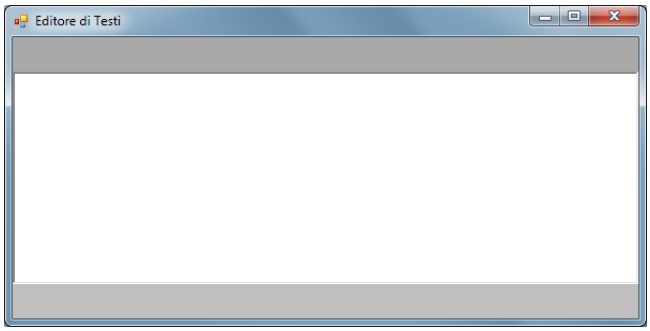
Un controllo RichTextBox serve per visualizzare e gestire del testo con formattazione. Il controllo ammette l'input da parte di un utente (che può scrivere, modificare, copiare, tagliare, incollare e cancellare parti di testo) per ottenere un contenuto e un aspetto diverso.

Il controllo può essere gestito in fase di progettazione mediante le sue proprietà, per approntare il suo aspetto iniziale, può essere modificato in visuale dall'utente e può essere modificato run-time da metodi appositi. In particolare si suggerisce di agganciare un menu contestuale al controllo, per agire sopra.

Infine il controllo offre numerosi metodi e proprietà che consentono di gestire il contenuto e l'aspetto in modo semplice.

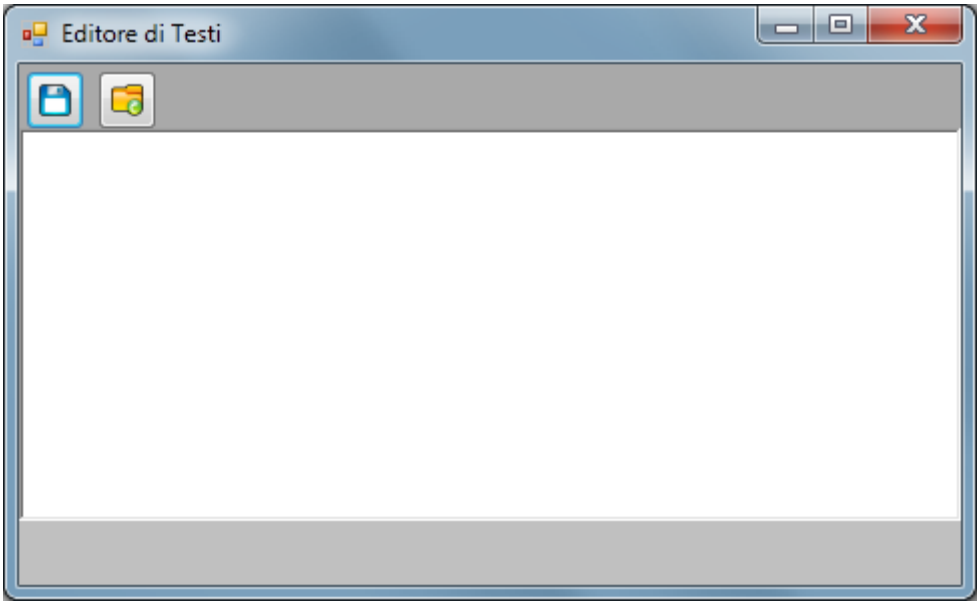
PROGETTO GUIDATO

EDITORE DI TESTI ELEMENTARE



- Crea un nuovo progetto visuale e salvalo
- Metti un componente Panel1 nel Form1 e imposta le seguenti proprietà:
  - **Dock** ← Top
  - **Height** ← 32
  - **BackColor** ← Silver
- Metti un componente Panel2 nel Form1 e imposta le seguenti proprietà:
  - **Dock** ← Bottom
  - **Height** ← 32
  - **BackColor** ← Silver
- Metti un componente RichTextBox1 nel Form1 e imposta le seguenti proprietà:
  - **Dock** ← Fill
- Metti un componente Button1 dentro il Panel1 (in alto) e imposta le seguenti proprietà:
  - **Height** ← 30
  - **Width** ← 30
  - **Text** ← (cancella tutto e lascia una stringa vuota)
  - **Image** ← importa una nuova immagine e cercane una come questa  grande 16pixel

- Metti un componente Button2 dentro il Panel1 (in alto) e imposta le seguenti proprietà:
  - **Height** ← 30
  - **Width** ← 30
  - **Text** ← (cancella tutto e lascia una stringa vuota)
  - **Image** ← importa una nuova immagine e cercane una come questa  grande 16pixel
- La figura seguente mostra un esempio di come dovrebbe risultare l'applicazione:

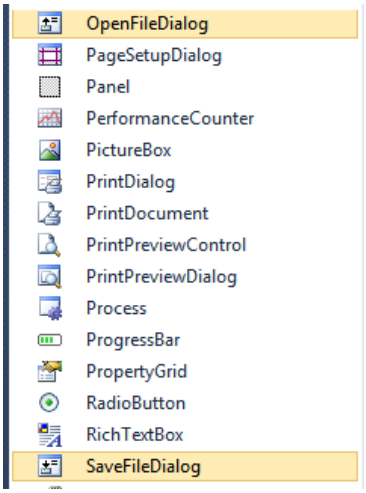


- Metti un componente SaveFileDialog1 dentro il Form1
- Poiché è un componente non visuale comparirà sotto il Form1, in un'apposita area per questi componenti
- Associa al Pulsante Button1 (salva) il seguente gestore:

VC#

```
//clic su Salva
saveFileDialog1.Filter = "documentoRTF|*.rtf|documentoTXT|*.txt";
saveFileDialog1.FilterIndex = 0;
saveFileDialog1.Title = "Salva il documento con un nome";
saveFileDialog1.FileName = "mioDocumento";
saveFileDialog1.DefaultExt = "rtf";
// Eseguo il dialogo e memorizzo la scelta dell'utente
DialogResult risposta = saveFileDialog1.ShowDialog();
// Se l'utente ha premuto OK
if (risposta == System.Windows.Forms.DialogResult.OK)
   // Se il nome del file non è una stringa vuota
   if (saveFileDialog1.FileName != "")
   {
       // Salvataggio
       richTextBox1.SaveFile(saveFileDialog1.FileName);
       MessageBox.Show("Salvato");
     };
```

- Metti un componente OpenFileDialog1 dentro il Form1
- Anche questo è un componente non visuale e comparirà sotto il Form1, nell'apposita area
- Associa al Pulsante Button2 (apri) il seguente gestore:



VC#

```
//clic su Apri
openFileDialog1.Filter = "documentoRTF|*.rtf|documentoTXT|*.txt";
openFileDialog1.FilterIndex = 0;
openFileDialog1.Title = "Apri un documento";
openFileDialog1.FileName = "";
openFileDialog1.DefaultExt = "rtf";
// Eseguo il dialogo e memorizzo la scelta dell'utente
DialogResult risposta = openFileDialog1.ShowDialog();
// Se l'utente ha premuto OK
if (risposta == System.Windows.Forms.DialogResult.OK)
// Se il nome del file non è una stringa vuota
if (openFileDialog1.FileName != "")
{
// Salvataggio
richTextBox1.LoadFile(openFileDialog1.FileName);
};
```

COMPONENTE SAVEFILEDIALOG

COMPONENTE SAVEFILEDIALOG

Il componente SaveFileDialog è un oggetto per la gestione del salvataggio su file di altri elementi. Il componente permette di salvare testo, testo formattato, immagini e altri tipi di file. Il componente può essere messo dentro il Form in fase di progettazione oppure può essere creato «al volo» come un qualsiasi oggetto.

PROPRIETÀ DEL COMPONENTE

Il componente SaveFileDialog, oltre alle consuete proprietà comuni a tutti i componenti, possiede diverse proprietà specifiche per la gestione del salvataggio su file.

Filter

La proprietà Filter (filtro) serve per elencare le estensioni ammesse dal dialogo. La proprietà è una stringa che elenca coppie del tipo «Nome utente»|«estensione». Per esempio:

```
saveFileDialog1.Filter = "documentoRTF|*.rtf|documentoTXT|*.txt";
```

indica che il componente ha due tipi nel filtro, il primo chiamato arbitrariamente documentoRTF che ha estensione rtf; il secondo chiamato arbitrariamente documentoTXT che ha estensione txt.

FilterIndex

La proprietà FilterIndex (indice del filtro) serve per indicare l'estensione di default tra quelle in elenco. La proprietà è un numero intero non negativo. Per esempio:

```
saveFileDialog1.FilterIndex = 0;
```

indica che il componente propone il primo filtro in elenco.

Title

La proprietà Title (titolo) è il titolo della finestra di dialogo. La proprietà è una stringa. Per esempio:

```
saveFileDialog1.Title = "Salva il documento con un nome";
```

indica che il titolo del dialogo è la frase riportata.

FileName

La proprietà FileName (nome del file) è il nome del file proposto nella finestra di dialogo. Può essere impostata prima di mostrare il dialogo per proporre all'utente un nome di default; l'utente può modificarlo e scriverne uno diverso. La proprietà è una stringa. Per esempio:

```
saveFileDialog1.FileName = "mioDocumento";
```

indica che il nome del file da salvare è la parola riportata.

DefaultExt

La proprietà DefaultExt (estensione di default del file) è l'estensione di default proposto nella finestra di dialogo. Può essere impostata prima di mostrare il dialogo per proporre all'utente un nome di default; l'utente può modificarlo e sceglierne uno diverso dall'elenco. La proprietà è una stringa (senza il punto). Per esempio:

```
saveFileDialog1.DefaultExt = "rtf";
```

indica che il tipo del file da salvare è la parola riportata.

METODI DEL COMPONENTE

ShowDialog()

Il metodo ShowDialog() è il principale metodo del componente.

Il metodo serve per mostrare la finestra di dialogo a video, con le proprietà indicate sopra. Il metodo propone una finestra modale, ovvero non è possibile proseguire con l'applicazione che l'ha lanciata senza prima chiudere la finestra di dialogo.

L'utente può chiudere la finestra di dialogo in diversi modi, ma generalmente la chiude premendo Salva oppure Annulla. Il metodo restituisce un valore che indica il modo con cui è stata chiusa la finestra di dialogo. Il valore restituito è di tipo `DialogResult` . Per esempio:

```
DialogResult risposta = saveFileDialog1.ShowDialog();
```

prima visualizza la finestra di dialogo, ma poi memorizza nella variabile risposta il modo con cui si è chiusa la finestra di dialogo.

Il tipo `DialogResult` elenca come attributi statici i valori ammessi come risposta che sono (potenzialmente) i seguenti:

Valore	Descrizione
None	Niente è restituito dalla finestra di dialogo. Generalmente significa che la finestra di dialogo modale prosegue.
OK	Il valore restituito della finestra di dialogo è OK (generalmente si preme OK, ma per questo componente significa che si è premuto **Salva**).
Cancel	Il valore restituito della finestra di dialogo è Cancel (in genere inviato da un pulsante **Annulla**).
Abort	Il valore restituito della finestra di dialogo è Abort (in genere inviato da un pulsante per interrompere, in questo caso chiudendo la finestra).
Retry	Il valore restituito della finestra di dialogo è Retry (in genere inviato da un pulsante Riprova).
Ignore	Il valore restituito della finestra di dialogo è Ignore (in genere inviato da un pulsante Ignora).
Yes	Il valore restituito della finestra di dialogo è Yes (in genere inviato da un pulsante SI).
No	Il valore restituito della finestra di dialogo è No (in genere inviato da un pulsante con l'etichetta NO).

Per esempio il codice seguente:

VC#

```
if (risposta == System.Windows.Forms.DialogResult.OK)
{
    MessageBox.Show("Salvato");
}
```

Controlla se il valore di risposta coincide con il valore costante «OK» e in tal caso mostra a video la frase "Salvato".

COMPONENTE OPENFILEDIALOG

COMPONENTE OPENFILEDIALOG

Il componente OpenFileDialog è un oggetto per la gestione dell'apertura di file. Il componente permette di aprire file di testo semplice, testo formattato, immagini e altri tipi di file. Il componente può essere messo dentro il Form in fase di progettazione oppure può essere creato «al volo» come un qualsiasi oggetto.

PROPRIETÀ DEL COMPONENTE

Il componente OpenFileDialog possiede proprietà e metodi in modo analogo al componente SaveFileDialog, oltre alle consuete proprietà comuni a tutti i componenti.

Per i fini di questo corso si fa riferimento solo a quanto già discusso per il componente SaveFileDialog: (Filter, FilterIndex, Title, FileName, DefaultExt); inoltre si userà il metodo ShowDialog, che restituisce gli stessi valori già discussi per SaveFileDialog.

GESTIONE DEI FILE COL RICHTEXTBOX

Il controllo RichTextBox possiede diversi metodi e proprietà già discussi in precedenza. Ma in particolare, in riferimento ai file, sono interessanti due metodi:

- **SaveFile** salvataggio del testo su file;
- **LoadFile** caricamento di testo da file;

IL METODO SAVEFILE

Il controllo RichTextBox dispone di un metodo SaveFile che consente di salvare su file il testo contenuto nelle proprietà Lines del controllo. Il metodo offre tre diversi modi in overload:

- **SaveFile(String)**
  Salva il contenuto del RichTextBox in un file (RTF) RTF;
- **SaveFile(String, RichTextBoxStreamType)**
  Salva il contenuto del RichTextBox in un tipo specifico di file.
- **SaveFile(Stream, RichTextBoxStreamType)**
  Salva il contenuto del RichTextBox in un flusso di dati aperto;

Ai fini del corso ci possiamo limitare al primo modo d'uso di SaveFile.

Il metodo SaveFile consente di salvare l'intero contenuto del controllo in un file con formattazione RTF (la cui estensione è generalmente .rtf) che è utilizzabile anche da altri programmi come Microsoft Word e WordPad. Se il nome file passato al parametro **path** è già presente nella directory specificata, il file verrà sovrascritto senza preavviso (distruggendo il preesistente). Per questo il controllo dell'esistenza del file è spesso affidata a codice utente o al dialogo SaveFileDialog.

Per esempio:

VC#
```
richTextBox1.SaveFile(saveFileDialog1.FileName);
```

ha l'effetto di salvare il testo del controllo `richTextBox1` in un file il cui nome è indicato dalla proprietà **saveFileDialog1.FileName** , con estensione automatica rtf.

La seconda versione del metodo SaveFile consente di specificare il tipo del file in cui salvare il contenuto del controllo `richTextBox1`. È consigliabile utilizzare questa funzionalità per assicurarsi che il file venga salvato nel formato appropriato in base al contenuto del controllo. Per esempio, se il documento non ha caratteristiche nello stile del carattere o nella sua colorazione, è possibile salvare il file come file di testo ASCII (testo non formattato) impostando il parametro **fileType** con **RichTextBoxStreamType.PlainText**.

Per esempio:

VC#
```
richTextBox1.SaveFile("pippo" , RichTextBoxStreamType.PlainText);
```

ha l'effetto di salvare il testo del controllo `richTextBox1` in un file il cui nome è «pippo» e il cui tipo è txt.

I tipi ammessi da **RichTextBoxStreamType** sono:

RichText	**Un file (o un flusso) con formato RTF, ovvero testo con formattazione.** **Questo tipo di testo ammette colori, font, grandezze diverse e qualità (grassetto, corsivo, sottolineato) specifico anche per ciascun carattere.**
PlainText	Un file (o un flusso) di testo normale senza formattazione, con spazi invece di oggetti OLE.
RichNoOleObjs	Un file (o un flusso) (RTF) del formato RTF con spazi anziché di oggetti OLE.
TextTextOleObjs	Un file (o un flusso) (RTF) del formato RTF con elementi di tipo oggetti OLE.
UnicodePlainText	Un flusso di testo contenente spazi invece che oggetti OLE. Il testo viene codificato in Unicode.

IL METODO LOADFILE

Il controllo RichTextBox dispone di un metodo LoadFile che consente di caricare da un file il testo che andrà contenuto nella proprietà Lines del controllo. Il metodo offre tre diversi modi in overload:

- **`LoadFile(String)`**
  Carica il contenuto da un file (RTF) RTF;
- **`LoadFile(String, RichTextBoxStreamType)`**
  Carica il contenuto da un tipo specifico di file.
- **`LoadFile(Stream, RichTextBoxStreamType)`**
  Carica il contenuto da un flusso di dati aperto;

Ai fini del corso ci possiamo limitare al solo primo modo d'uso di LoadFile.

Il metodo LoadFile consente di caricare l'intero contenuto del controllo da un file con formattazione RTF (la cui estensione è generalmente .rtf) che è riproducibile anche da altri programmi come Microsoft Word e WordPad. Se il nome file passato al parametro **`path`** non è già presente nella directory specificata, si solleva un eccezione di errore (file inesistente); per questo il controllo dell'esistenza del file è spesso affidata a codice utente o al dialogo OpenFileDialog.

Per esempio:

VC#
```
richTextBox1.LoadFile(openFileDialog1.FileName);
```

ha l'effetto di caricare il testo del controllo `richTextBox1` da un file il cui nome è indicato dalla proprietà **`openFileDialog1.FileName`** , con estensione automatica rtf.

Gli altri overload del metodo sono trascurati, ai fini del presente corso.

ESERCIZI

ESERCIZI

PROGETTO «EDITORE DI TESTI»

CONTROLLO DI MODIFICA DEL TESTO

- Prepara un applicazione come quella proposta nell'esercizio guidato precedente;
- Usa una variabile logica (`bool`) per stabilire se il testo del controllo è modificato (da salvare) oppure no;
- Quando l'utente tenta di chiudere la finestra, ma il testo è stato modificato, il programma deve avvertire l'utente e chiedergli se vuole salvare il file oppure lasciarlo inalterato

SALVA VS SALVA CON NOME

- Usa una variabile di testo per memorizzare il nome del file da elaborare; se il file è nuovo (applicazione appena lanciata) il file non ha ancora un nome;
- Modifica l'applicazione precedente prevedendo due pulsanti differenti:
  - Salva, che salva il file col nome già stabilito (tranne che al primo salvataggio);
  - Salva con nome, che salva una copia del file in un altro file.

GESTIRE FINESTRE MULTIPLE

- Se hai già studiato le finestre multiple MDI, costruisci un'applicazione che gestisce più file contemporaneamente.

INSERIRE MENU PRINCIPALI

- Se hai già studiato i menu, costruisci un'applicazione con i menu File e Modifica;
- Prova anche i menu per il carattere e il paragrafo.

INSERIRE MENU CONTESTUALI

- Se hai già studiato i menu contestuali, costruisci un'applicazione con i menu contestuali;
- Inserisci le voci di menu per il carattere e il paragrafo.

ESPERIMENTI CON LE LISTBOX

- Verifica se il controllo ListBox possiede i metodi SaveFile() e LoadFile().
- Prova a ipotizzare come gestire una classifica elencata con un ListBox e come salvarla su un file.

SOMMARIO