

Generare dei files in base ad un particolare tracciato (quarta parte)

http://www.vbsimple.net/activity/act_11_4.htm

Richiesta di: [Roberto Corona](#) - 12 Gennaio 2001

Difficoltà: 5 / 5

Ho bisogno di scrivere un programmino che, dopo aver analizzato dei files contenenti dei particolari tracciati, estragga dei dati da un foglio di Excel e tramite questi generi dei files di testo con una particolare struttura determinata dal file del tracciato.

[<< Continua dalla parte 3](#)

```
50. Private Sub ListaSezioni_Click()  
51.     Dim TMPSEZIONE As TIPOSEZIONE  
52.     If ListaSezioni.ListIndex < 0 Then Exit Sub  
53.  
54.     Riga2Sezione SEZIONI.Item(ListaSezioni.ListIndex + 1), TMPSEZIONE  
55.     FileTracciato.SelStart = TMPSEZIONE.INIZIO  
56.     FileTracciato.SelLength = TMPSEZIONE.FINE - TMPSEZIONE.INIZIO + 1  
57.     ListaCampi.ListIndex = Val(TMPSEZIONE.CAMPO)  
58.     RigaFissaCheck.Value = TMPSEZIONE.RIGAFISSA  
59.     LunghezzaFissaCheck.Value = TMPSEZIONE.LUNGHEZZAFISSA  
60.     AllineaSinistraCheck.Value = TMPSEZIONE.ALLINEAMENTO  
61.     FillerText.Text = TMPSEZIONE.FILLER  
62. End Sub  
63.
```

Nel momento in cui l'utente clicca sul nome di una sezione dalla *ListBox* **ListaSezioni** saranno mostrati i dati che tale sezione contiene.


Il click sopra un elemento estrarrà la sezione corrispondente dalla *Collection* **SEZIONI** (riga 54). Da tale sezione saranno letti, uno per uno, i valori al fine di impostare i controlli dell'interfaccia in modo da farli corrispondere con i dati della sezione (righe 55-61).

```
64. Private Sub PulsanteEliminaSezione_Click()  
65.     Dim CONTA As Integer  
66.     Dim TMPSEZIONE As TIPOSEZIONE  
67.     If ListaSezioni.ListIndex < 0 Then Exit Sub  
68.     ListaSezioni_Click  
69.     FileTracciato.SelColor = RGB(0, 0, 0)  
70.     SEZIONI.Remove ListaSezioni.ListIndex + 1  
71.     MostraSezioni  
72.     ListaSezioni.ListIndex = SEZIONI.Count - 1  
73.     FileTracciato.SelLength = 0  
74.     FileTracciato.SetFocus  
75. End Sub  
76.
```

Non avrebbe molto senso dare all'utente la possibilità di inserire nuove sezioni ma non poterle eliminare dalla *Collection* delle sezioni.

Così, nel momento del click sul pulsante **PulsanteEliminaSezione** viene, innanzitutto, ricliccata la sezione dalla ListBox ListaSezioni (riga 68), in modo da ottenere la sezione evidenziata sulla *RichTextBox* **FileTracciato**. Ottenuto questo sarà ripristinato il colore nero originario (riga 69), sarà eliminata la sezione dalla Collection (riga 70) e saranno nuovamente mostrate le sezioni tramite la Sub MostraSezioni (riga 71).

```
77. Private Sub FileTracciato_SelChange()  
78.     Dim CONTA As Integer  
79.     Dim TMPSEZIONE As TIPOSEZIONE  
80.     Dim CONTROLLA As Boolean  
81.     Dim INIZIADA As Integer  
82.     Dim FINEA As Integer
```

Quella che vediamo è la parte più difficile e delicata del progetto. Nel momento in cui viene spostato il cursore del testo sopra la *RichTextBox*  **FileTracciato** o viene effettuata una selezione del testo tramite mouse o tastiera, scatta l'[evento](#) *SelChange*.

Una particolare cura deve essere data a questa routine poiché il posizionamento del cursore attraverso d'essa può richiamare nuovamente questa routine, generando un ciclo continuo fino all'esaurimento dell'area di [stack](#).

Il controllo della posizione si basa su due parametri: la coordinata del cursore ed il colore del testo selezionato. Infatti, un testo già colorato in rosso indica che il cursore si trova all'interno di una sezione o il testo selezionato è intersecato con una sezione.

```
83.     CONTROLLA = True  
84.     PulsanteAggiungiSezione.Enabled = True  
85.     PulsanteEliminaSezione.Enabled = False  
86.     If FileTracciato.SelColor = 0 Then CONTROLLA = False
```

A questo scopo utilizzeremo una variabile di nome **CONTROLLA**, che inizialmente sarà impostata a True. Nel momento in cui viene controllato il colore del testo selezionato, se esso è nero la variabile sarà impostata a False (riga 86).

Cosa fa esattamente questa variabile?

Se il testo selezionato è **tutto** nero, sappiamo con certezza che a quelle non ci sono sezioni; se il testo selezionato, in qualche parte è rosso, ci troviamo nelle vicinanze di una sezione.

```
87.     If (CONTROLLA = True) And (TROVAPOSIZIONE = False) Then  
88.         INIZIADA = FileTracciato.SelStart  
89.         FINEA = INIZIADA + FileTracciato.SelLength  
90.         For CONTA = 1 To SEZIONI.Count  
91.             Riga2Sezione SEZIONI.Item(CONTA), TMPSEZIONE  
92.             If (INIZIADA >= TMPSEZIONE.INIZIO) And (INIZIADA <= TMPSEZIONE.FINE)  
Then Exit For  
93.             If (FINEA >= TMPSEZIONE.INIZIO) And (FINEA <= TMPSEZIONE.FINE) Then  
Exit For  
94.             If (TMPSEZIONE.INIZIO <= FINEA) And (TMPSEZIONE.INIZIO >= INIZIADA)  
Then Exit For  
95.         Next CONTA  
96.         If CONTA > SEZIONI.Count Then Exit Sub
```

Alla riga 87 viene verificato se la variabile **CONTROLLA** è True (ovvero il cursore si trova all'interno di una sezione oppure il testo selezionato copre una sezione. Altresì viene controllato che la variabile **TROVAPOSIZIONE** sia False (ovvero il cursore è stato spostato dall'utente e non dal codice).

Quando questa situazione si verifica sarà necessario controllare tutte le sezioni per vedere quale sezione si trova a quelle coordinate.

Saranno memorizzate le coordinate iniziali e finali della selezione nelle variabili INIZIADA e FINEA (righe 88 e 89). Dopodiché inizierà la scansione di tutte le sezioni mediante l'estrazione di una sezione per volta dall'array SEZIONI (riga 91).

Se la selezione è all'interno della sezione analizzata (riga 92) oppure la parte finale della selezione si trova all'interno della sezione (riga 93), oppure la parte iniziale della selezione si trova all'interno del testo selezionato (riga 94), viene interrotto il ciclo di scansione delle sezioni.

L'interruzione del ciclo fa sì che la variabile CONTA contenga l'indice della sezione che si interseca con il testo selezionato. Se il ciclo non è stato interrotto, alla sua uscita, la variabile CONTA avrà un valore superiore al numero complessivo di sezioni. In quest'ultimo caso non è stata trovata la sezione corrispondente e la routine termina qui, senza evidenziare alcuna sezione.

```
97.         TROVAPOSIZIONE = True
98.         FileTracciato.SelStart = TMPSEZIONE.INIZIO
99.         FileTracciato.SelLength = TMPSEZIONE.FINE - TMPSEZIONE.INIZIO + 1
100.        ListaCampi.ListIndex = Val(TMPSEZIONE.CAMPO)
101.        RigaFissaCheck.Value = TMPSEZIONE.RIGAFISSA
102.        LunghezzaFissaCheck.Value = TMPSEZIONE.LUNGHEZZAFISSA
103.        AllineaSinistraCheck.Value = TMPSEZIONE.ALLINEAMENTO
104.        FillerText.Text = TMPSEZIONE.FILLER
105.        ListaSezioni.ListIndex = CONTA - 1
106.        TROVAPOSIZIONE = False
107.        PulsanteAggiungiSezione.Enabled = False
108.        PulsanteEliminaSezione.Enabled = True
109.    End If
110. End Sub
111.
```

Il raggiungimento della riga 97 presuppone che sia stata trovata la sezione intersecata con il testo selezionato.

A tale scopo possiamo effettuare la lettura dei parametri della sezione **TMPSEZIONE**. Poiché ci occorrerà spostare il cursore per mostrare all'utente la posizione e l'ampiezza della sezione, impostiamo la variabile **TROVAPOSIZIONE** a True e alla riga successiva spostiamo il cursore all'inizio della sezione ed impostiamo la selezione fino alla fine della sezione. Questo fa sì che scatti nuovamente l'evento SelChange.

Se non avessimo impostato la variabile globale **TROVAPOSIZIONE** a True, si sarebbe innescato un circolo vizioso che richiama continuamente questo evento per posizionare il cursore, fino a generare un errore di stack esaurito.

Questo problema di [ricorsività](#) viene risolto dalla variabile **TROVAPOSIZIONE**; infatti alla riga 87, prima di effettuare l'operazione di ricerca (e riposizionamento) devono essere validate due condizioni: il colore lo stato di **TROVAPOSIZIONE**. L'impostazione della variabile al valore True fa sì che non venga richiamato l'evento SelChange due o più volte.

Evidenziata la sezione sarà possibile visualizzare anche tutte le altre proprietà della sezione (righe 98-105), prima di ripristinare lo stato di **TROVAPOSIZIONE** a False.

```

112. Private Sub PulsanteCreaDefinizione_Click()
113.     Dim NOMEFILE As String
114.     Dim FILEOK As Boolean
115.     Dim FILENR As Integer
116.     Dim CONTA As Integer
117.     On Error Resume Next
118.     While FILEOK = False
119.         Err.Clear
120.         FILEOK = False
121.         NOMEFILE = InputBox("Inserisci il nome del file in cui salvare", "Salva
definizione")
122.         If NOMEFILE = "" Then Exit Sub
123.         If FileEsiste(NOMEFILE) = False Then FILEOK = True
124.         If FILEOK = True Then
125.             FILENR = FreeFile
126.             Open NOMEFILE For Output As FILENR
127.             If Err.Number <> 0 Then FILEOK = False
128.         End If
129.     Wend
130.     Write #FILENR, "Fibia Data Stripper"
131.     For CONTA = 1 To SEZIONI.Count
132.         Write #FILENR, SEZIONI.Item(CONTA)
133.     Next CONTA
134.     Close FILENR
135.     MsgBox "Salvataggio effettuato!", vbInformation + vbOKOnly, "Crea definizione"
136. End Sub
137.

```

Il click sopra il pulsante CreaDefinizione effettua il salvataggio della definizione in un file FDF. Alla riga 121 viene richiesto il nome del file in cui salvare; di tale file sarà controllata l'esistenza. Se l'utente ha premuto il tasto Annulla la routine terminerà (riga 122). Nel caso che il file inserito non esistesse sarà possibile salvare poiché la variabile **FILEOK** conterrà il valore True. Appurato che il file non esiste, verrà tentata la creazione (riga 126); nel caso che il disco sia protetto o pieno, si genererà un errore che verrà intercettato dalla riga successiva (riga 127) la quale imposterà il valore di **FILEOK** a False.

Il ciclo While (righe 118-129) durerà fintanto che FILEOK non conterrà il valore True, ovvero il file non esiste e la scrittura sul disco è permessa.

Alla riga 130 viene scritta l'intestazione "*Fibia Data Stripper*" che contraddistinguerà i files FDF.

Solo adesso sarà possibile scrivere le sezioni all'interno del file per il salvataggio".

```


138. Private Sub PulsanteCaricaDefinizione_Click()
139.     Dim FILENR As Integer
140.     Dim BUFFER As String
141.     Dim TMPSEZIONE As TIPOSEZIONE
142.     BUFFER = InputBox("Seleziona il nome della definizione da caricare", "Carica
definizione")
143.     If FileEsiste(BUFFER) = False Then Exit Sub
144.     FILENR = FreeFile
145.     Open BUFFER For Input As FILENR
146.     Input #FILENR, BUFFER
147.     If UCase(Left(BUFFER, 19)) <> "FIBIA DATA STRIPPER" Then Exit Sub
148.     Me.Enabled = False
149.     While SEZIONI.Count > 0
150.         SEZIONI.Remove 1
151.     Wend
152.     ListaSezioni.Clear
153.     TROVAPOSIZIONE = True
154.     FileTracciato.SelStart = 0
155.     FileTracciato.SelLength = Len(FileTracciato.Text) + 1
156.     FileTracciato.SelColor = RGB(0, 0, 0)

```

```
157.      While Not EOF(FILENR)
158.          Input #FILENR, BUFFER
159.          SEZIONI.Add BUFFER
160.          Riga2Sezione BUFFER, TMPSEZIONE
161.          FileTracciato.SelStart = TMPSEZIONE.INIZIO
162.          FileTracciato.SelLength = TMPSEZIONE.FINE - TMPSEZIONE.INIZIO + 1
163.          FileTracciato.SelColor = RGB(255, 0, 0)
164.      Wend
165.      TROVAPOSIZIONE = False
166.      Close FILENR
167.      Me.Enabled = True
168.      MostraSezioni
169. End Sub
170.
```

Il click sopra il pulsante **CaricaDefinizione** richiede l'inserimento del nome del file FDF da caricare (riga 142). Ovviamente tale file deve necessariamente esistere. Se esso non esiste la routine terminerà qui.

Avendo la sicurezza che tale file esiste possiamo leggere l'intestazione del file per verificare che esso sia un file valido (riga 146). Un'intestazione valida fa sì che la funzione termini qui.

Il file richiesto è valido! È possibile caricarlo, ma prima sarà necessario svuotare la *Collection* **SEZIONI** e la *ListBox*  **ListaSezioni** (righe 149-152) per riempirle nuovamente. Ma prima di riempirle nuovamente togliamo tutte le parti in rosso del testo (righe 153-156) tenendo la variabile di blocco **TROVAPOSIZIONE** impostata a True.

Alle righe 157-164 vengono lette le sezioni dal file FDF, sono scritte nella *Collection* **SEZIONI** e man mano viene colorato il testo in rosso dove abbiamo una sezione.

Prima di uscire dalla routine ripristiniamo lo stato normale del form reimpostando la variabile **TROVAPOSIZIONE** a False e ridisegnando le sezioni mediante la Sub **MostraSezioni**.

```
171. Private Sub PulsanteChiudi_Click()
172.     Set SEZIONI = Nothing
173.     Me.Hide
174. End Sub
```

Lo studio del form si conclude con il pulsante **PulsanteChiudi** che azzera la *Collection* **SEZIONI** e nasconde il form, ritornando il controllo al form chiamante (*MainForm*).

[Segue parte 5 >>](#)

[Fibia FBI](#)
27 Gennaio 2001



[Torna all'introduzione delle Richieste dei lettori](#)
