



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

http://www.vbsimple.net/activity/act_11_2.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 1](#)

Per sviluppare questo progetto utilizzeremo il controllo **RichTextBox** e **DAO 3.5** per accedere al foglio di Excel. Pertanto è necessario selezionare il controllo **Microsoft Rich Textbox Control** dalla voce Componenti del menu **Progetto**. È anche necessario inserire un riferimento a **Microsoft DAO 3.5 Object Library** tramite la voce Riferimenti del menu **Progetto**.

Il progetto si compone di due **forms** ed un **modulo** standard. Quest'ultimo conterrà due variabili globali, un nuovo tipo di dati di nome **TIPOSEZIONE** ed alcune funzioni che utilizzeremo parecchie volte all'interno del nostro progetto.

Nei due form invece avremo gli strumenti minimi per definire la struttura del tracciato ed il form principale, molto semplice, per la scelta dei files da utilizzare.

Prima di addentrarci nei forms vediamo il contenuto del nostro modulo:

```
1. Option Explicit
2.
3. Public Type TIPOSEZIONE
4.     INIZIO As Integer
5.     FINE As Integer
6.     CAMPO As Integer
7.     RIGAFISSA As Integer
8.     LUNGHEZZAFISSA As Integer
9.     ALLINEAMENTO As Integer
10.    FILLER As String * 1
11. End Type
12.
13. Public FILEXLS As DAO.Database
14. Public FOGLIOXLS As DAO.Recordset
15.
```

Alla riga 3 definiamo il tipo di dati **TIPOSEZIONE**. In sostanza una sezione è un segmento del file tracciato con una posizione iniziale e una finale, associato ad un campo del foglio Excel e con dei dati aggiuntivi che stabiliscono se la sezione è fissa o si ripete

per ogni record, se la lunghezza è fissa o regolata dall'ampiezza del campo associato. Nel caso sia fissa è necessario definire l'allineamento che i dati devono avere (sinistra o destra) ed il carattere di riempimento per raggiungere la lunghezza desiderata.

Così il tipo dati **TIPOSEZIONE** si compone di 7 campi:

- **INIZIO**
Definisce la posizione iniziale della sezione all'interno del file tracciato
- **FINE**
Definisce la posizione finale della sezione
- **CAMPO**
Indica l'ordine del campo associato del foglio XLS
- **RIGAFISSA**
Indica in maniera numerica se la riga è fissa o si ripete per ogni riga del file XLS
- **LARGHEZZAFISSA**
Indica in maniera numerica se la larghezza della sezione è variabile, determinata dal campo, oppure è
- **ALLINEAMENTO**
Indica, nel caso di larghezza fissa della sezione, se i dati devono essere allineati a sinistra o a destra
- **FILLER**
Indica, in caso di larghezza fissa, quale carattere deve essere utilizzato per riempire gli spazi fino a raggiungere la larghezza della sezione

Alle righe 13 e 14 abbiamo definito due variabili globali. La prima è **FILEXLS** ed indica il database Excel per i dati da estrarre. La seconda variabile è il *Recordset* **FOGLIOXLS** che conterrà i dati da estrarre dal database FILEXLS.

Seguono, nel modulo, un paio di funzioni che utilizzeremo più avanti.

```
16. Public Function FileEsiste(ByVal NOMEFILE As String) As Boolean
17.     On Error GoTo ERRORE
18.     Dim FILENR As Integer
19.     FileEsiste = False
20.     FILENR = FreeFile
21.     Open NOMEFILE For Input As FILENR
22.     Close FILENR
23.     FileEsiste = True
24.     Exit Function
25. ERRORE:
26.     FileEsiste = False
27. End Function
28.
```

La funzione FileEsiste viene spiegata approfonditamente nell'[HowTo dedicato alla verifica se un dato file esiste](#).

```
29. Public Function Sezione2Riga(ByRef SEZIONE As TIPOSEZIONE) As String
30.     Sezione2Riga = SEZIONE.INIZIO & ";"
31.     Sezione2Riga = Sezione2Riga & SEZIONE.FINE & ";"
32.     Sezione2Riga = Sezione2Riga & SEZIONE.CAMPO & ";"
33.     Sezione2Riga = Sezione2Riga & SEZIONE.RIGAFISSA & ";"
34.     Sezione2Riga = Sezione2Riga & SEZIONE.LUNGHEZZAFISSA & ";"
35.     Sezione2Riga = Sezione2Riga & SEZIONE.FILLER & ";"
36.     Sezione2Riga = Sezione2Riga & SEZIONE.ALLINEAMENTO
37. End Function
```

38.

La funzione **Sezione2Riga** serve per trasformare una sezione di tipo TIPOSEZIONE in una stringa. Questo perché le singole sezioni saranno tenute all'interno di una [Collection](#); con Visual Basic 5, tuttavia, non è possibile inserire elementi di tipi definiti dall'utente all'interno di una Collection. A tale scopo, nel momento in cui inseriremo i dati nella Collection convertiremo la sezione in una stringa ed inseriremo quella.

La funzione **Sezione2Riga** unisce i singoli campi della sezione in una stringa separandoli temite un punto e virgola.

Segue un'altra funzione che effettua l'operazione inversa, cioè converte una stringa in una sezione di tipo TIPOSEZIONE.

```

39. Public Sub Riga2Sezione(ByVal RIGA As String, ByRef SEZIONE As TIPOSEZIONE)
40.     Dim TMPSEZIONE As String
41.
42.     TMPSEZIONE = RIGA
43.     SEZIONE.INIZIO = Mid(TMPSEZIONE, 1, InStr(1, TMPSEZIONE, ";") - 1)
44.     TMPSEZIONE = Mid(TMPSEZIONE, Len(CStr(SEZIONE.INIZIO)) + 2)
45.
46.     SEZIONE.FINE = Mid(TMPSEZIONE, 1, InStr(1, TMPSEZIONE, ";") - 1)
47.     TMPSEZIONE = Mid(TMPSEZIONE, Len(CStr(SEZIONE.FINE)) + 2)
48.
49.     SEZIONE.CAMPO = Mid(TMPSEZIONE, 1, InStr(1, TMPSEZIONE, ";") - 1)
50.     TMPSEZIONE = Mid(TMPSEZIONE, Len(CStr(SEZIONE.CAMPO)) + 2)
51.
52.     SEZIONE.RIGAFISSA = Mid(TMPSEZIONE, 1, InStr(1, TMPSEZIONE, ";") - 1)
53.     TMPSEZIONE = Mid(TMPSEZIONE, Len(CStr(SEZIONE.RIGAFISSA)) + 2)
54.
55.     SEZIONE.LUNGHEZZAFISSA = Mid(TMPSEZIONE, 1, InStr(1, TMPSEZIONE, ";") - 1)
56.     TMPSEZIONE = Mid(TMPSEZIONE, Len(CStr(SEZIONE.LUNGHEZZAFISSA)) + 2)
57.
58.     SEZIONE.FILLER = Mid(TMPSEZIONE, 1, InStr(1, TMPSEZIONE, ";") - 1)
59.     TMPSEZIONE = Mid(TMPSEZIONE, Len(CStr(SEZIONE.FILLER)) + 2)
60.
61.     SEZIONE.ALLINEAMENTO = TMPSEZIONE
62. End Sub
63.

```

La funzione effettua la ricerca del punto e virgola ed inserisce i dati all'interno della sezione [passata per riferimento](#).

L'ultima Sub del modulo è **OrdinaSezioni** e, come dice il nome stesso, effettua l'ordinamento delle sezioni in base al punto di inizio della sezione.

Essa richiede come parametro la Collection contenente gli elementi da ordinare, sempre passata per riferimento, affinché sia possibile scrivere gli elementi una volta ordinati.

Il metodo di ordinamento è il classico Bubble sort (ordinamento a bolle), approfondito nella sezione [Informazioni aggiuntive](#).

```

64. Public Sub OrdinaSezioni(ByRef SEZIONI As Collection)
65.     Dim ARRAYSEZIONI() As String
66.     Dim CONTA As Integer
67.     Dim CONTA2 As Integer
68.     Dim TMPSEZIONE As TIPOSEZIONE
69.     Dim SUCCSEZIONE As TIPOSEZIONE
70.     ReDim ARRAYSEZIONI(5)

```

Alla riga 65 abbiamo dichiarato un'array ([matrice](#)) di stringhe di nome **ARRAYSEZIONI**

senza specificare la sua ampiezza. Questo perché vogliamo utilizzare un dimensionamento dinamico effettuato in [fase di esecuzione](#).

Infatti abbiamo il primo ridimensionamento alla riga 70 impostando la dimensione della matrice a 5.

Alle righe 68 e 69 abbiamo dichiarato due variabili di tipo **TIPOSEZIONE**. Esse saranno utilizzate per estrarre due valori dalla *Collection* e confrontarli per l'ordinamento.

```
71.      For CONTA = 1 To SEZIONI.Count
72.          If UBound(ARRAYSEZIONI) < CONTA Then ReDim Preserve ARRAYSEZIONI(UBound
              (ARRAYSEZIONI) + 5)
73.          ARRAYSEZIONI(CONTA) = SEZIONI.Item(CONTA)
74.      Next CONTA
```

Come prima operazione della Sub eseguiamo una copia dei dati della Collection SEZIONI nella matrice ARRAYSEZIONI.

Non conoscendo a priori la dimensione della Collection abbiamo voluto effettuare il ridimensionamento della matrice a 5 elementi per volta.

Infatti, quando la dimensione dell'array è minore dell'elemento in corso di estrazione, viene incrementata la dimensione dell'array di 5 elementi, preservando il contenuto dell'array.

Esistono modi più semplici ed efficienti per effettuare questa operazione, ma abbiamo voluto sfruttare l'occasione per mostrare il funzionamento del dimensionamento dinamico.

Alla riga 73 abbiamo l'inserimento dell'elemento dalla Collection alla matrice.

```
75.      For CONTA = 1 To SEZIONI.Count - 1
76.          For CONTA2 = CONTA + 1 To SEZIONI.Count
77.              Riga2Sezione ARRAYSEZIONI(CONTA), TMPSEZIONE
78.              Riga2Sezione ARRAYSEZIONI(CONTA2), SUCCSEZIONE
79.              If TMPSEZIONE.INIZIO > SUCCSEZIONE.INIZIO Then
80.                  ARRAYSEZIONI(0) = ARRAYSEZIONI(CONTA2)
81.                  ARRAYSEZIONI(CONTA2) = ARRAYSEZIONI(CONTA)
82.                  ARRAYSEZIONI(CONTA) = ARRAYSEZIONI(0)
83.                  ARRAYSEZIONI(0) = ""
84.              End If
85.          Next CONTA2
86.      Next CONTA
```

In questo ciclo abbiamo l'operazione di sorting (ordinamento) tramite Bubble sort. L'elemento 0 della matrice serve per effettuare lo scambio dei dati tra due celle dell'array.

```
87.      CONTA2 = SEZIONI.Count
88.      While SEZIONI.Count > 0
89.          SEZIONI.Remove 1
90.      Wend
91.      For CONTA = 1 To CONTA2
92.          SEZIONI.Add ARRAYSEZIONI(CONTA)
93.      Next CONTA
94. End Sub
```

Effettuato il sorting sarà necessario rimettere tutti i dati nella Collection in maniera ordinata. Per questo alla riga 87 salviamo la dimensione della Collection nella variabile CONTA2.

Avendo messo in salvo la dimensione originale, possiamo tranquillamente svuotare la Collection (righe 88-90) e poi riempirla nuovamente utilizzando la dimensione salvata precedentemente (righe 91-93).

[Segue parte 3 >>](#)

[Fibia FBI](#)

27 Gennaio 2001



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