di Francesco Petroni

IL NUOVO VISUAL BASIC FOR APPLICATION

Questo articolo fa parte della serie dedicata da MC al nuovo Office della Microsoft, l'Office 97. Una delle novità più evolute presenti nel sistema Office 97 è l'ambiente di programmazione, che si chiama Visual Basic for Application, versione 5.0, e che serve, come noto, per sviluppare applicazioni, più o meno chiuse, basate sui componenti Office.

Il lungo cammino verso il VBA 5.0

Nei precedenti appuntamenti abbiamo parlato delle funzionalità generali, valide per tutto l'ambiente Office, dell'integrazione tra Office 97 ed Internet/Intranet, dei singoli componenti della famiglia, quella normale e quella allargata.

Queste note, relative al Visual Basic for Application, sono state preparate utilizzando Office versione Inglese, ma quando leggerete quest'articolo sarà uscita la versione Italiana, che peraltro già trovate nella prova che appare in altre pagine di questo stesso numero di MC.

Mi sembra doveroso spendere qualche parola sulla storia del VBA e quindi sugli strumenti per la programmazione che hanno caratterizzato le versioni precedenti dei vari componenti di Office 97.

All'inizio, ogni prodotto "andava per conto suo". Le prime versioni di Excel (dalla 2 alla 4) disponevano del solo linguaggio Macro, che si appoggiava su tipi particolari di foglio, le prime versioni di Word (dalla 1 alla 6) disponevano di un loro linguaggio WordBasic, le prime versioni di Access (dalla 1 alla 2) disponevano di un loro linguaggio AccessBasic. PowerPoint non era per nulla programmabile.

Sia chiaro che le versioni citate permettevano comunque di sviluppare applicazioni, anche molto evolute. I linguaggi non erano allineati tra di loro (indimenticabili le traduzioni in italiano dei comandi Macro di Excel!), e la programmazione interprodotto era abba-

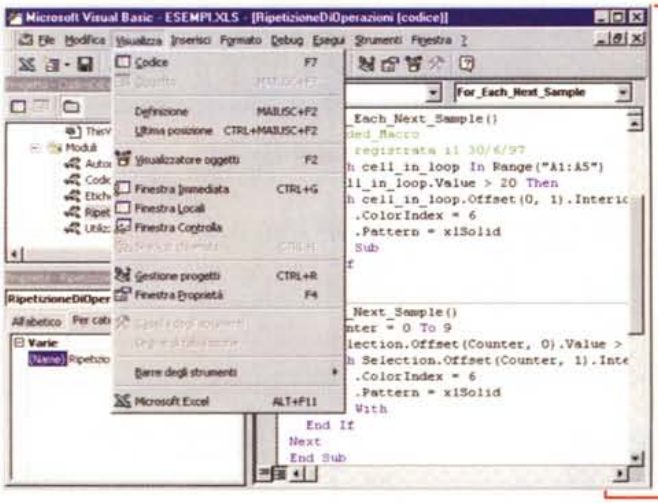


Figura 1 - VBA 5.0 - L'Ambiente Integrato di Sviluppo IDE.
Se, da un qualsiasi componente della Suite Office 97, si esegue il comando Tools, Macro, Editor Visual Basic, si entra in questo ambiente operativo multifinestra, nel quale si costruisce l'applicazione, scrivendo o editando il codice, disegnando Forms, e nel quale si controlla l'esecuzione delle routine sfruttando i numerosi strumenti per il Debug. Qui vediamo l'ambiente VBE della versione italiana di Office, anche se lavoreremo con la versione originale.

stanza ostica.

Con la prima versione per Windows 95 di Office c'è stato un primo avvicinamento tra i vari componenti di Office, grazie al consolidamento della tecnologia OLE Automation, che è una della caratteristiche fondamentali del VBA. Tramite OLE Automation ogni componente della famiglia Office adotta un "Modello ad Oggetti", e quindi può essere programmato anche "dal di fuori".

In altre parole un'applicazione VBA e OLE Automation, qualsiasi sia il prodotto di partenza, può manipolare oggetti di qualsiasi altro prodotto, sfruttandone le librerie. In Office 95 anche PowerPoint adottava il modello ad Oggetti (e diventava quindi programmabile dal di fuori). Word, invece, rimaneva un po' indietro. Poteva essere programmato ma veniva visto come un unico grosso oggetto. Ricordo agli interessati all'argomento la serie di articoli dedicati al VBA apparsi sui numeri più recenti di MC (ad esempio il 168).

Con Office 97 è finalmente avvenuto l'aggancio totale tra tutti i componenti. Indichiamo, schematicamente e per punti, cosa questo significhi:

- tutti i componenti standard (Excel, Word, PowerPoint ed Access) adottano un modello ad oggetti,
- tutti i componenti sono VBA compatibili,
- da ciascuno di essi può essere attivato l'ambiente di sviluppo (IDE) che è una funzionalità condivisa (ne parliamo subito dopo),
- l'ambiente IDE comprende anche un Form Editor con il quale si disegnano le Finestre che servono nell'applicazione,
- da ciascuno di essi può essere attivata la funzionalità di registrazione delle Macro,
- ciascuno di essi può sfruttare, tramite VBA, le varie funzionalità condivise da VBA, ad esempio il motore DAO,
- ciascuno di essi può sfruttare componenti ActiveX.

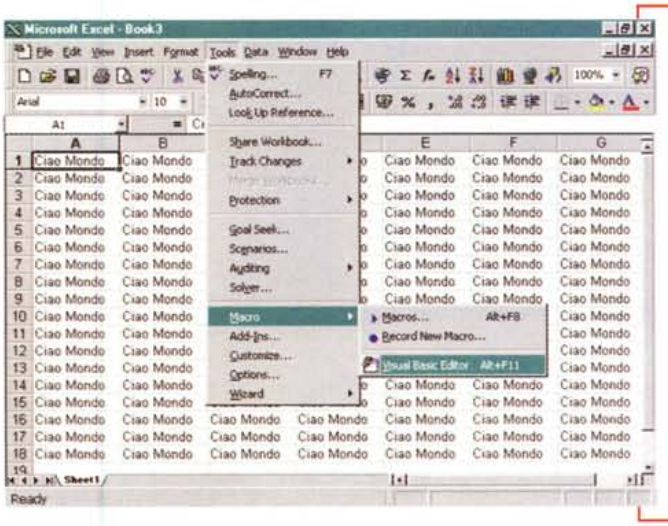


Figura 2 - VBA 5.0 - Registrazione di una semplice Macro - Effetto finale. Anche in Office 97 è utilizzabile il comodo strumento operativo Registratore di Macro, che serve sia a chi deve semplicemente memorizzare sequenze di operazioni, sia a chi deve realizzare applicazioni complesse e vuole, tramite il registratore, intercettare quale siano le proprietà di un oggetto, oppure quali siano i comandi con i quali si imposta una determinata proprietà. L'uso del Registratore di Macro è comunque il sistema più corretto per avvicinarsi a questo tipo di programmazione.

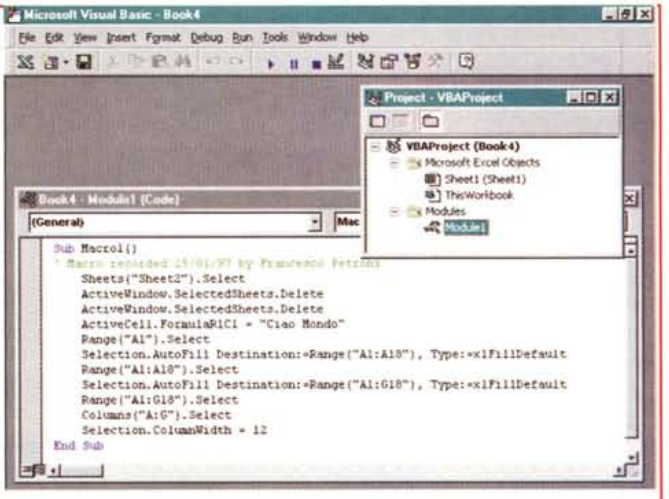


Figura 3 - VBA 5.0 - Registrazione di una semplice Macro - Il Codice sottostante. Terminata la registrazione e spento il registratore si può passare nel Visual Basic Editor (VBE), che, come abbiamo già visto nella prima figura, è un ambiente abbastanza articolato. Molto importante è la finestra Project Explorer che mostra, in una vista ad albero, gli oggetti "in giro" nell'applicazione. Poiché per registrare siamo partiti da un Workbook con tre fogli e ne abbiamo, con la Macro, eliminati due, ci ritroviamo l'oggetto Workbook e un solo oggetto Sheet1. Poi abbiamo un solo Modulo che è quello in cui sono stati riportati i comandi VBA con i quali si scrive qualche cosa nella prima cella e la si copia in tutte le celle che appaiono sul video.

DAO, la mia fissazione

Sono un po' fissato con DAO e quindi utilizzeremo la tecnologia di Accesso ai Dati, quella che applica il modello ad oggetti anche al Database (DAO: Data Access Object), per alcuni esperimenti eseguiti con i vari componenti. Realizzeremo delle procedure che fanno tre cose:

1 - attivano una Form nella quale indicare una serie di criteri di selezione,

2 - sulla base dei Criteri impostati eseguono un'estrazione di dati da un Database,

3 - riversano sul prodotto di destinazione i dati risultanti dalla estrazione.

Ebbene: i primi due passi vengono eseguiti usando le identiche istruzioni di programmazione VBA e DAO. Il terzo passo richiede una serie di istruzioni che dipendono anche dall'oggetto su

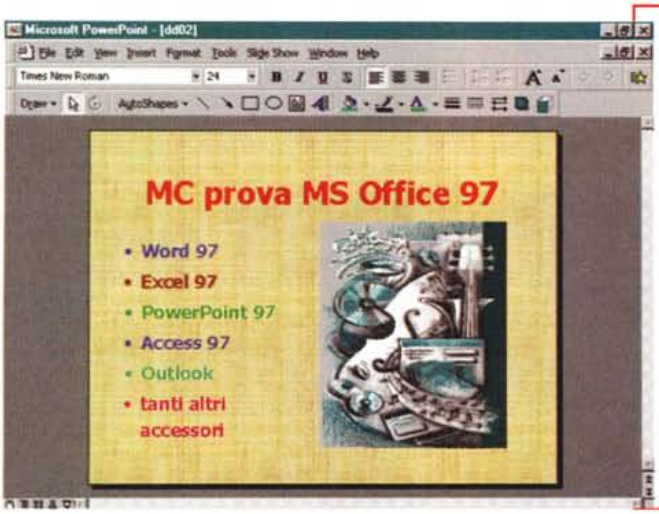


Figura 4 - VBA 5.0 - L'Arte dell'individuazione degli Oggetti e delle Proprietà - Risultato finale. Questa è una Macro registrata partendo da PowerPoint e da una situazione vuota, con nessuna slide. È stata quindi inserita una nuova Slide di tipo appropriato, ne è stato digitato il Titolo, è stata selezionata la Shape con l'elenco puntato, sono state inserite le varie voci dell'elenco, è stato letto e posizionato un file grafico, è stato scelto un tipo di riempimento per lo sfondo della Slide. Questo è il risultato.

cui i dati vengono riversati. I singoli dati estratti si possono quindi riversare:

- in una cella di un foglio Excel,
- in una riga di un documento Word,
- in una tabella di Access che in tal modo viene alimentata,
- in una slide PowerPoint.

IDE Integrated Development Environment

Per entrare direttamente nell'ambiente di sviluppo integrato IDE, partendo da qualsiasi componente Office (escluso Access), si deve eseguire il comando Tools, Macro, Visual Basic Editor (oppure, in alternativa, la scorciatoia Alt+F11).

L'ambiente presenta una serie di finestre che vanno aperte, chiuse, dimensionate, disposte a seconda delle necessità (figura 1). Le Finestre sono:

- ProjectExplorer, che mostra l'elenco degli oggetti disponibili nel progetto che si sta sviluppando,
- Code, l'editor del codice, che finalmente dispone di una serie di strumenti di aiuto per chi sta scrivendo il codice,
- ObjectBrowser, la finestra che serve per scegliere l'oggetto desiderato in maniera guidata e per verificarne la sintassi,
- UserForm, quando il progetto prevede di utilizzare una Form, che va disegnata sfruttando questo apposito Editor grafico,
- PropertyWindow, la finestra (al solito richiamabile con il tasto F4) che esplicita le proprietà dell'oggetto selezionato.

Esistono poi una serie di finestre per il Debug: Immediate Windows, Locals Windows, Watch Windows (in figura 1 ne vediamo le traduzioni in italiano).

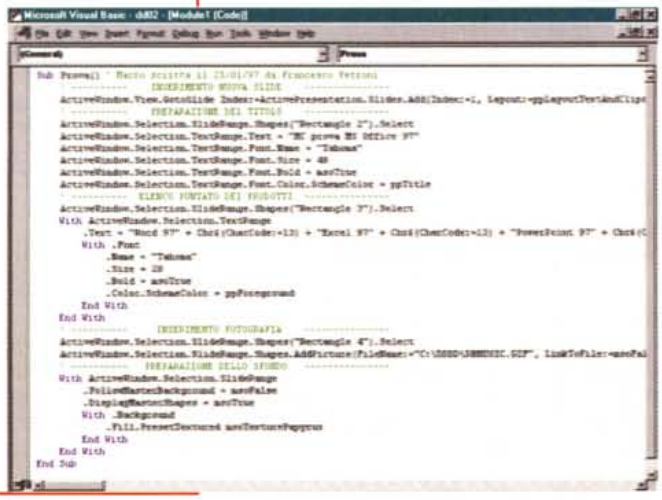
Come detto, in caso di necessità, usando il Form Editor, è possibile creare una Form sulla quale possono essere inseriti tutti gli oggetti standard di Windows. Quando si lavora con il Form Editor l'ambiente diventa del tutto analogo a quello del Visual Basic 5.0. C'è un Pannello con i vari Controlli, selezionato uno dei controlli si può attivare la sua Finestra delle Proprietà, facendo un doppio click sul controllo si attiva la finestra Code nella quale scrivere le routine legate agli eventi sul controllo stesso.

Come in Visual Basic si possono utilizzare controlli aggiuntivi, che ora si chiamano ActiveX e che hanno sempre la desinenza OCX.

L'organizzazione dell'ambiente, basato su così tante finestre, il cui numero aumenta se si usano ad esempio più Forms, è un po' incasinata, per cui risulta comodo, sia pure solo in fase di sviluppo, utilizzare una modalità video SuperVGA.

Anche il Visual Basic 5.0, prossimo venturo, utilizzerà lo stesso ambiente.

Figura 5 - VBA 5.0 - L'Arte dell'individuazione degli Oggetti e delle Proprietà - Il Codice sottostante. Fermata la registrazione si passa all'Editor Visual Basic per verificare il codice trascritto. C'è da dire che in generale questo codice, prodotto dal registratore di Macro, è molto prolisso. Ad esempio inserisce molte istruzioni che non producono nessun effetto e che quindi si possono tranquillamente eliminare. Anche per motivi fotografici (volevamo rimanere in una sola videata) abbiamo ridotto il codice ad una ventina di righe, mentre all'inizio erano oltre sessanta.



L'approccio più corretto

Il sistema più facile per avvicinarsi al VBA è quello di utilizzare il registratore di Macro, realizzando dapprima semplici operazioni, per poi controllare il codice che è stato trascritto automaticamente.

Successivamente si può provare a cambiare parti del codice, o per alleggerirlo o per modificare l'effetto della Macro. Vi proponiamo due esperimenti, il primo con Excel (in figure 2 e 3) ed il secondo con PowerPoint (figure 4 e 5).

Facciamo insieme il primo.

Accendiamo il registratore, poi scriviamo una frase nella prima cella, copiamola in giù e a destra fino a coprire tutta parte visibile del foglio (da A1 a G18). Poi selezioniamo le colonne riempite, allarghiamole e poi spegniamo il registratore.

Eliminiamo le prime sette colonne del foglio per svuotarlo di qualsiasi contenuto e rieseguiamo la macro.

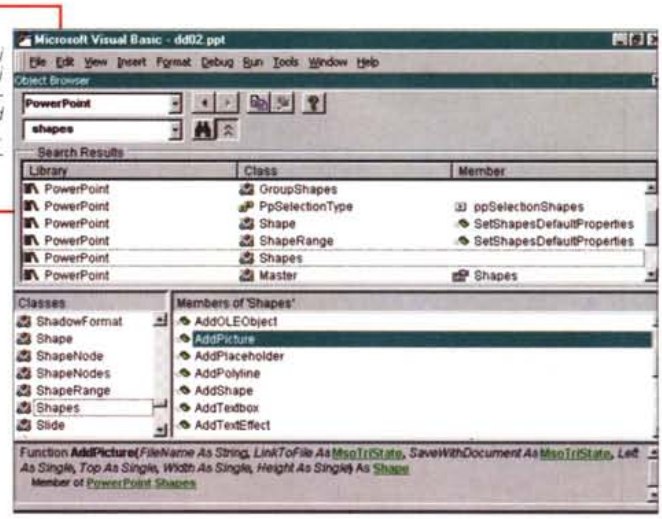


Figura 6 - VBA 5.0 - L'Object Browser (in italiano Visualizzatore Oggetti). Il Registratore di Macro è, come detto, uno strumento indispensabile sia per chi deve semplicemente memorizzare una serie di operazioni ripetitive sia per chi sviluppa applicazioni importanti. Per costui è molto utile la finestra Object Browser che permette di selezionare una Libreria, di indicare una parola chiave (ad esempio un Oggetto, oppure una Proprietà, ecc.) e di cercare tutte le attinenze. Una volta trovata l'istruzione, la funzione, la proprietà, ecc. viene fornita la sintassi esatta per la sua applicazione.

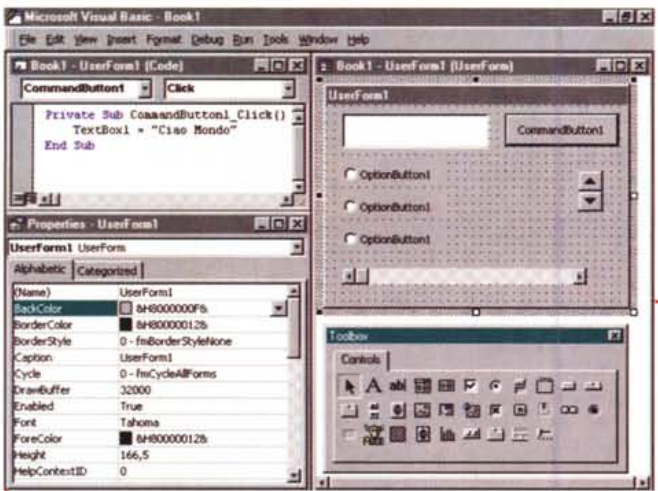


Figura 7 - VBA 5.0 - L'Editor delle Form Parte importante dell'IDE è l'Editor delle Form. In pratica si usa quando, nell'applicazione che si sta sviluppando, occorra inserire una Dialog Box per la digitazione, da parte dell'utente dell'applicazione, di una serie di dati necessari per la sua corretta esecuzione. La Form si disegna sfruttando uno speciale Editor che mostra la Form e che consente, tramite una serie di Toolbar, di inserirvi Controlli di vario genere, per ognuno dei quali vanno impostate le varie proprietà. Ci sono poi istruzioni per attivare e per disattivare, al momento opportuno, la Box e per passare dati verso e dalla Box stessa.

Il risultato lo vediamo in figura 2, mentre il codice lo vediamo in figura 3. È del tutto comprensibile, nel senso che ad ogni operazione eseguita corrisponde un comando del codice ben individuabile.

Una possibile variante, che vi suggeriamo, è quella di utilizzare un'istruzione di Input che serva per digitare in una Box la frase da inserire nella prima cella:

```
X = InputBox("Digitare la Frase da Copiare")
ActiveCell.FormulaR1C1=X
```

Altra possibilità è quella di scrivere direttamente:

```
Cells(1,1) = InputBox("Digitare la Frase da Copiare")
```

Queste due "varianti" non possono essere realizzate con il registratore in quanto utilizzano comandi non corrispondenti a comandi normali registrabili.

Si può ora analizzare il ProjectBrowser che mostra una ramificazione che vede al primo livello il VBA Project, immediatamente sotto troviamo l'oggetto Workbook che a sua volta contiene i vari Sheet. Pari livello, rispetto al Workbook, sono i Moduli e le Forms, che a loro volta sono contenitori di singoli individui.

Passiamo al secondo esperimento, eseguito con PowerPoint.

Lanciamo PowerPoint e chiudiamo qualsiasi presentazione. In questa situazione non si può fare pressoché nulla di operativo ma si può iniziare a registrare una Macro.

Inseriamo una nuova Slide, scegliendo un tipo che mostri un Titolo in alto, un Elenco Puntato a sinistra e un'Immagine a destra. Digitiamo il titolo, poi gli elementi dell'elenco, ed inseriamo l'immagine sfruttando il comando Inserisci File. Poi selezioniamo la Slide, per la quale impostiamo un motivo di riempimento di nostro gradimento. Infine spegniamo il registratore.

I rapporti con l'ambiente di programmazione

Se ci rimettiamo nella situazione di partenza e lanciamo la Macro appena registrata (Tools, Macro, Macros, Run) questa riesegue in un attimo tutte le operazioni che abbiamo precedentemente eseguito (in figura 4).

Se passiamo nell'Editor per esaminare il codice trascritto (in figura 5) possiamo notare alcune cose:

- ogni slide appartiene ad un tipo identificato dalla proprietà Layout,
- ogni slide contiene un certo numero di Shapes, che identificano i suoi vari componenti e che vanno selezionate per intervenire sul singolo componente,
- in generale la sintassi di un comando VBA di PowerPoint è molto complessa per il fatto che è complessa l'identificazione di ciascun elemento,
- in generale il codice prodotto dal Registratore di Macro è molto prolisso per il fatto che vengono registrate anche operazioni e impostazioni non eseguite.

Ad esempio, se si imposta una qualsiasi delle proprietà relative al Font di un scritta, vengono comunque registrate tutte le proprietà relative alla scritta stessa. Inoltre il Registratore tende ad usare il comando With.

Ad esempio, supponiamo di aver im-

postato la proprietà grassetto per un Titolo, il codice risultante è:

```
With ActiveWindow.Selection.TextRange
   .Text = "MC prova MS Office 97"
   With .Font
        .Name = "Tahoma"
        .Size = 44
        .Bold = msoFalse
        .Italic = msoFalse
        .Underline = msoFalse
        .Shadow = msoFalse
        .Emboss = msoFalse
        .BaselineOffset = 0
        .AutoRotateNumbers = msoFalse
        .Color.SchemeColor = ppTitle
   End With
End With
```

Ma bastava semplicemente:

```
With ActiveWindow.Selection.TextRange
   .Text = "MC prova MS Office 97"
   With .Font
        .Name = "Tahoma"
        .Size = 44
   End With
End With
```

Oppure, rinunciando ai servizi del comando With:

```
ActiveWindow.Selection.TextRange.Text = "MC prova MS Office 97"
ActiveWindow.Selection.TextRange.Font.Name = "Tahoma"
ActiveWindow.Selection.TextRange.Font.Size = 44
```

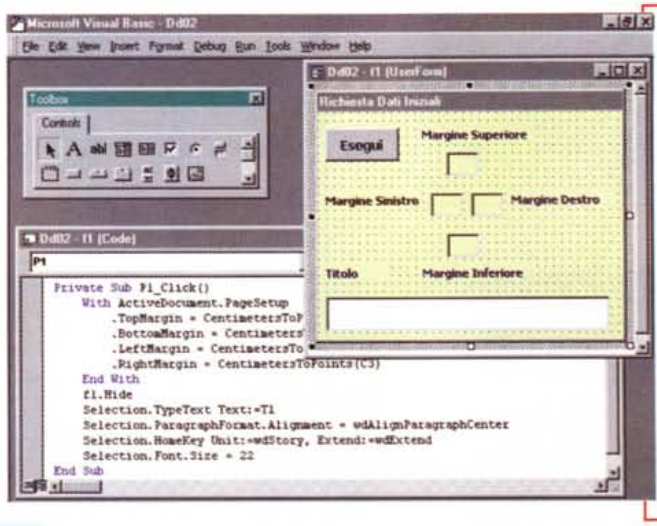


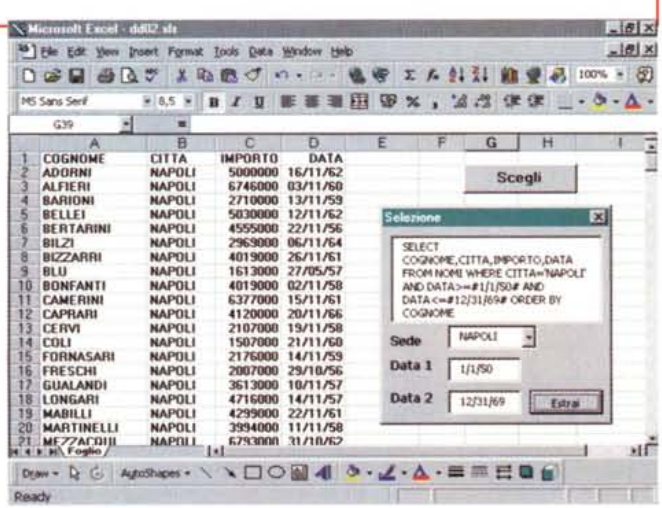
Figura 8 - VBA 5.0 - Una Macro per personalizzare per l'impostazione della pagina di Word.
Questa è una prima e semplice applicazione di quanto detto. Si tratta di una microprocedura che va lanciata quando si stia lavorando con un documento Word. Abbiamo disegnato una Box nella quale si inseriscono quattro valori numerici, indicanti le dimensioni dei quattro margini della pagina, e una stringa, che viene usata come titolo del documento. Come si vede, il codice necessario è, in questo caso, molto compatto.

Come si vede il listato prodotto con la registrazione di Macro può essere molto manipolato a parità di risultato raggiunto. Oppure ancora, volendo ricorrere alla istruzione With, che mette a "fattor comune" pezzi di programma:

```
With ActiveWindow.Selection.TextRange
        .Text = "MC prova MS Office 97"
        .Font.Name = "Tahoma"
        .Font.Size = 44
End With
```

Chi invece volesse scrivere a mano le varie istruzioni deve combattere con una sintassi non tanto complicata da

Figura 9 - VBA 5.0 - Excel 97 - Una Form per creare una Query che attiva il motore DAO - Risultato finale.
I sistemi per scaricare, in un Foglio di Excel, dati provenienti da un Database esterno sono tantissimi. Nell'esempio che abbiamo realizzato sfruttiamo la Libreria DAO 3.5, disponibile con Office 97. In pratica utilizziamo una Box nella quale inseriamo alcuni criteri per la selezione dei dati che vogliamo scaricare. Il meccanismo è questo: sul foglio si clicca su Scegli, appare la Box che va riempita con i criteri desiderati. Con il pulsante Estrai posto sulla Box si manda in esecuzione l'estrazione dei dati e quindi il rimpimento delle celle del foglio. Ricordiamo che le righe del foglio di Excel 97 sono 65.536.



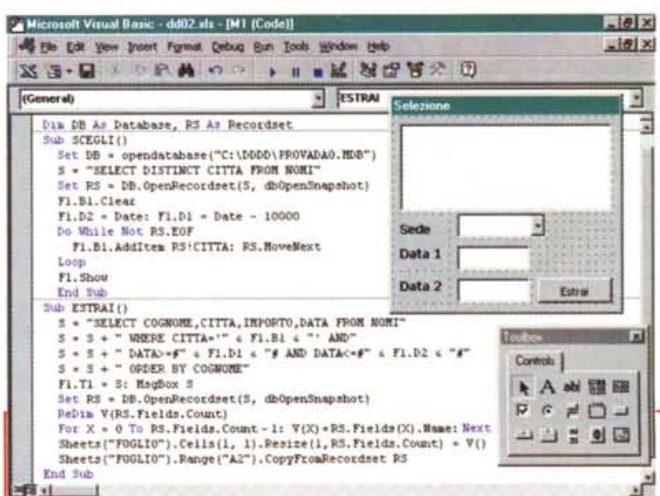


Figura 10 - VBA 5.0 - Excel 97 - Una Form per creare una Query che attiva il motore DAO - Il Codice sottostante.
Come detto nella precedente didascalia, il codice consiste sostanzialmente in due routines. La prima è quella che viene lanciata premendo il tasto Scelgi posto sul Foglio e che serve per preparare la Box, per inserirvi due date e per produrre, con una procedurina DAO, l'elenco delle CITTÀ presenti in tabella che a sua volta alimenta la ComboBox. La seconda routine viene lanciata direttamente dalla BOX, compone l'istruzione SQL, sulla base di quanto indicato nella Box stessa, ed esegue l'estrazione. Per scaricare il risultato dell'estrazione, Dati e Titoli dei Campi, si usano istruzioni DAO. Notare ancora una volta la potenza e la sinteticità del comando CopyFromRecordset.

capire quanto complessa da scrivere.

Non è complicata in quanto si basa sulla filosofia Object Based secondo la quale un'istruzione serve per impostare una proprietà di un Oggetto, ma è complessa perché in molti casi è complessa l'identificazione di un Oggetto o della Proprietà.

Aiuta nel lavoro di identificazione degli elementi lo strumento Visualizzatore Oggetti (l'Object Browser in figura 6) che esegue operazioni di ricerca degli elementi tra le varie librerie aperte e che mostra la loro corretta sintassi. Diventa produttivo solo per chi abbia già dimestichezza con l'ambiente e che quindi, quando cerca qualcosa, non fa troppi giri a vuoto.

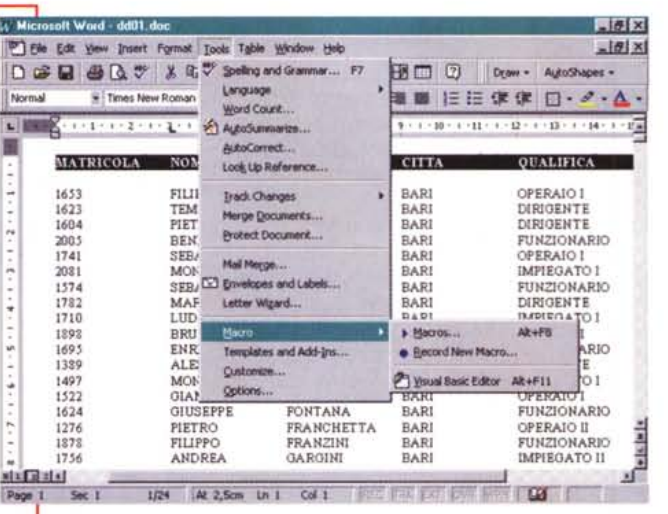


Figura 11 - VBA 5.0 - Word 97 - Una UserForm per creare una Query che attiva il motore DAO - Risultato finale.
Vediamo un documento Word il cui contenuto è costituito da una Tabella di Dati provenienti da un Database esterno. Il nostro obiettivo è quello di sfruttare le tecniche DAO, che, come diremo nell'articolo, vengono condivise da tutto l'ambiente VBA, per scaricare dati sulla pagina di Word. Con l'occasione vediamo anche il ramo di menu dal quale si possono eseguire le Macro, una volta realizzate.

Usiamo una Form personalizzata

Sempre, o quasi sempre, in un'applicazione occorre eseguire degli Input. Esistono, da sempre nei vari Visual Basic usciti fino ad ora, due semplici funzioni:

```
    X = MsgBox("Premi OK per proseguire, Annulla per interrompere",1)
    X = InputBox("Digitare la Sede desiderata","Richiesta Sede","ROMA")
```

Spesso l'input è più complesso, ad esempio quando occorra inserire più valori. In questo caso va disegnata una Form che contenga i vari oggetti adatti per l'immissione (TextBox, ComboBox, ListBox, CheckBox, OptionButton, ecc.). La Form va attivata e poi, eseguito il lavoro di immissione, va disattivata. In generale l'uscita dalla Form si esegue pigiando un pulsante posto sulla Form stessa. Schematizzando e ipotizzando che la Form si chiami F1:

```
...
preparazione dei dati per la Form
F1.Show              attivazione della Form
immissione dei dati e preparazione dei dati per uso successivo
F1.Hide              disattivazione della Form
...
```

Per disegnare la Form si usa un apposito Editor che possiamo ammirare in figura 7.

Un primo facile esperimento ve lo proponiamo in figura 8.

Si tratta di una Macro da lanciare da Word che attiva una Form che chiede cinque cose, i quattro valori dei margini della pagina e un titolo. Una volta chiusa la Form questi valori vengono utilizzati per impostare i margini della pagina del documento attivo e per inserire un titolo (scritta in alto al centro della pagina) nel documento stesso.

La Form, visibile in figura 8, contiene le quattro TextBox per i quattro margini (da C1 a C4) e una TextBox per il titolo (T1). Contiene poi un CommandButton che serve per lanciare il programma.

Il programma può essere costruito con il Registratore di Macro. Il codice prodotto va modificato in quanto vanno sostituiti i valori digitati "a mano" durante la registrazione con i valori degli elementi C1, C2, C3, C4, T1. Lo vediamo in figura 8, figura in cui non vediamo il codice che serve ad attivare la Form (in pratica basta l'istruzione F1.Show).

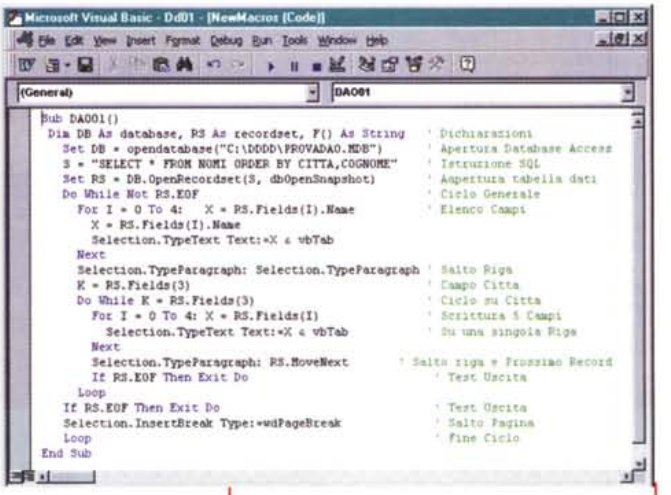


Figura 12 VBA 5.0 - Word 97 - Una UserForm per creare una Query che attiva il motore DAO - Il Codice sottostante
Questo è il codice che produce .. l'immagine precedente. Lo abbiamo un po' complicato in quanto abbiamo delegato al DAO l'ordinamento per il campo CITTA' in modo da poter creare un elenco di dati, raggruppati per CITTA' ed ordinati per COGNOME, con i titoli riportati per ogni nuova città ed un salto pagina alla fine di ciascuna città.

Due esperimenti con DAO

Proseguiamo con due esperimenti più nobili.

Vogliamo utilizzare due miniprocedure DAO che estraggono dati, da un Database in formato MDB, per river-

sarli su Excel (il primo esperimento) o su Word (il secondo esperimento).

Del primo esperimento vediamo, in figura 9, il comportamento sul foglio Excel, che spiega inequivocabilmente il funzionamento della procedura.

Dal tasto Scegli posto sul foglio si attiva la Form, nella quale l'utente sceglie due Date e una Sede. Sulla base delle scelte eseguite viene costruita una istruzione SQL che viene visualizzata nella TextBox e poi viene utilizzata per eseguire la query DAO.

In pratica ci sono solo due routine, la prima legata all'evento Click sul pulsante Esegui, posto sul Foglio, e la seconda legata al pulsante Estrai, posto sulla Form. Le vediamo in figura 10.

La seconda procedura lavora su Word, vediamo l'effetto in figura 11 ed il codice in figura 12. In pratica si usa DAO per eseguire una query che mette in ordine, per Città e per Cognome, una serie di nominativi.

Il Recordset prodotto con la query viene fatto scorrere tramite due cicli, uno che va dall'inizio a fine file e uno, interno, che scorre i Cognomi a parità di Città. Questo consente di inserire (a rottura di Città, direbbero i tecnici) un nuovo Titolo e di forzare un Salto Pagina ad ogni fine Città. La struttura del programma relativo allo scorrimento del Recordset RS è semplice:

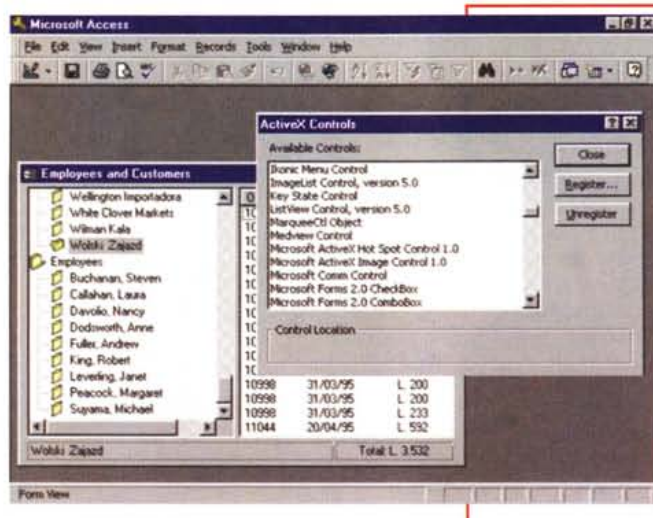


Figura 13 - VBA 5.0 - ActiveX (formerly OLE Objects). Abbiamo detto che VBA 5.0 dispone di una serie di funzionalità in comune. Abbiamo, ad esempio, appena visto come la tecnologia DAO sia sfruttabile sia da Excel che da Word. Un altro aspetto importantissimo è l'adozione della tecnologia ActiveX, che è, come noto, l'evoluzione della tecnologia OLE Controls. In pratica, qualsiasi sia il tipo di applicazione che si sta sviluppando, è possibile utilizzare i controlli aggiuntivi OCX, che sono, dal punto di vista pratico, delle librerie di oggetti che svolgono una serie di funzionalità utili per l'applicazione stessa.

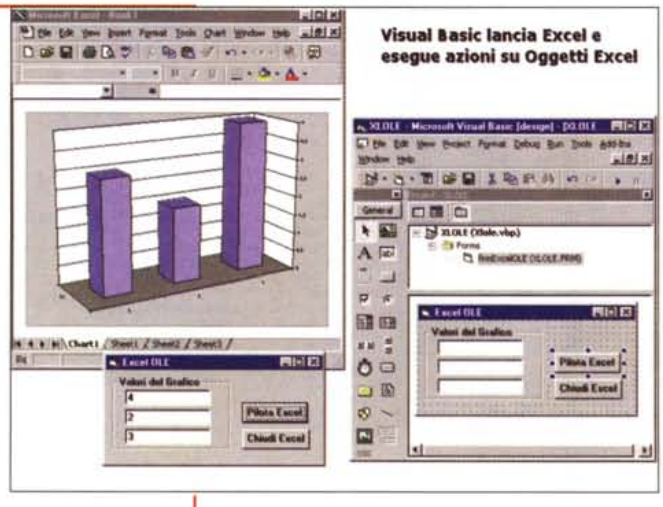


Figura 14 - VBA 5.0 - L'ulteriore possibilità: OLE Automation VBA 5.0 comprende e consolida la tecnologia OLE Automation, quella che consente di realizzare applicazioni che manipolano oggetti "esposti" dai vari componenti Office, indipendentemente da quale sia il componente di partenza e da quale sia il componente di destinazione. Abbiamo utilizzato una serie di esercizi presentati negli articoli, dedicati all'OLE Automation, pubblicati su MC n. 168 e precedenti. Hanno funzionato egregiamente a conferma del fatto che in definitiva VBA 5.0 è un'evoluzione dei VBA precedente.

```
Do While Not RS.Eof                    ' ciclo esterno
           K=RS.Fields(3)              ' il terzo campo è la Città
           Do While K=RS.Fields(3)          ' ciclo sulla Città
                    ...                     ' omissis
           RS.MoveNext                      ' scorrimento
           Loop                             ' fine ciclo interno
    Loop                                    ' fine ciclo interno
```

Come al solito, le istruzioni interne al Word, che servono per scrivere il dato e per eseguire il salto pagina, e cioè:

```
SelectionTypeText: Text:=X
SelectionInsertBreak Type:=wdPageBreak
```

le possiamo costruire accendendo, per pochi secondi, il registratore.

Conclusioni

Questi sono i principi alla base del VBA, ai quali si possono aggiungere altri principi più complessi.

Il primo è sicuramente la possibilità di sfruttare i Controlli ActiveX, ovvero altre librerie esterne che permettono di utilizzare nell'applicazione che si sta sviluppando oggetti più complessi (breve citazione in figura 13).

Il secondo è la possibilità di sfruttare la tecnologia OLE Automation. Questa è sostanzialmente la stessa presente nella versione precedente (Office 95 e più volte trattata in questa stessa rubrica) e la citiamo nella figura 14. Ne parleremo ancora.

MC

Chi ama ascoltare bene la musica, prima di qualsiasi acquisto ascolta da sempre una voce autorevole: quella di AUDIOREVIEW. Dalle sue pagine ogni mese uno staff di veri esperti dà ai lettori, esigenti o anche alle prime armi, ogni informazione e suggerimento per un ascolto migliore: chi la legge apprende ogni volta qualcosa di nuovo e importante. Prestando ascolto alle migliaia di prove, recensioni e notizie pubblicate in dodici anni, centinaia di migliaia di lettori hanno imparato a orientarsi nel vasto mercato dell'alta fedeltà e della musica, scegliendo bene fra impianti hi-fi, home theater, dischi e CD. E consultando gli aggiornamenti costanti dei prezzi di tutti i componenti hi-fi ed home theater hanno potuto acquistare il meglio, in linea con i consigli di AUDIOREVIEW, senza sbagliare mai.

... poi ho comprato AUDIOREVIEW.

technimedia
Pagina dopo pagina, le nostre passioni.

AUDIOREVIEW. Impianti senza rimpianti.