di Francesco Petroni

Microsoft Visual Basic 5.0 - Costruzione di ActiveX - Primi esercizi

Gli obiettivi di questo articolo sono due. Il primo è quello di esplorare le funzionalità presenti nel Microsoft Visual Basic 5.0 dedicate alla realizzazione di componenti attivi. Si tratta di funzionalità del tutto nuove, in quanto non erano presenti nella precedente versione di VB. Il secondo è quello di cercare di fare un po' di chiarezza attorno al termine ATTIVO, oggi diventato di moda e quindi associato un po' a tutto.
Si sente parlare di Componenti Attivi, di Controlli ActiveX, di Server Attivi, di Documenti Attivi, di Pagine Attive e di altro ancora. Gli ambiti in cui sono usati questi termini sono non solo Internet (il vero responsabile di tutto quello che sta succedendo), ma anche il Workgroup Computing e il Personal Computing. Parleremo soprattutto del Visual Basic 5.0, del quale in questo stesso numero presentiamo la prova. Anzi questo articolo costituisce un approfondimento di una delle novità più importanti del VB5, la possibilità di realizzare componenti ActiveX.

Piccolo riassunto delle versioni precedenti

Nella storia della Microinformatica il primo esempio concreto di riutilizzabilità di componenti è rappresentato proprio dal Visual Basic, la cui prima versione già sfruttava i controlli VBX. Lo sviluppatore VB poteva inserire nella propria applicazione controlli VBX che svolgessero funzionalità specifiche, che venivano in tal modo facilmente aggiunte all'applicazione. I controlli VBX hanno avuto un enorme successo anche per il fatto che sono stati adottati in molti altri ambienti di sviluppo. Sono sopravvissuti fino alla versione 3.0 del VB.
All'epoca la Microsoft mise a disposizione di chi voleva sviluppare propri componenti VBX, e quindi non voleva solamente utilizzarli, uno specifico strumento di sviluppo che si chiamava Control Developer's Kit (CDK), e che si basava comunque sull'uso del linguaggio C.
Successivamente, con la versione 4.0 di Visual Basic, i VBX sono stati sostituiti dai controlli OCX, basati sull'interfaccia OLE, e quindi più in linea con le tecnologie standard di Windows. Contemporaneamente sono stati aggiornati i vari CDK, che sfruttavano le versio-

ni via via più aggiornate del C, C++, del Visual C.
Il maggior difetto dei controlli OCX, che li rende praticamente inutilizzabili in applicazioni sviluppate per Internet, è costituito dal fatto che sono sempre molto voluminosi in quanto necessitano, per poter funzionare, del supporto di voluminose librerie specifiche, quali la MFC, la Microsoft Foundation Class. Gli ActiveX, che conservano la desinenza OCX, invece sono più leggeri (e quindi più facilmente scaricabili via Internet) dei vecchi controlli OCX, in quanto è stato ridotto il numero delle interfacce COM supportate. La Microsoft inoltre mette a disposizione degli sviluppatori strumenti alternativi per lo sviluppo di ActiveX, non più solo C e i suoi derivati, ma anche il Visual Basic 5, il Visual J++,e l'ActiveX Controls Framework (che però non è visuale).
Visual Basic 5 permette sia di sviluppare ActiveX sia, in quanto HOST, di utilizzare controlli ActiveX. Questa possibilità facilita la fase di sviluppo in quanto i nuovi ActiveX che possono essere testati direttamente "sul posto".

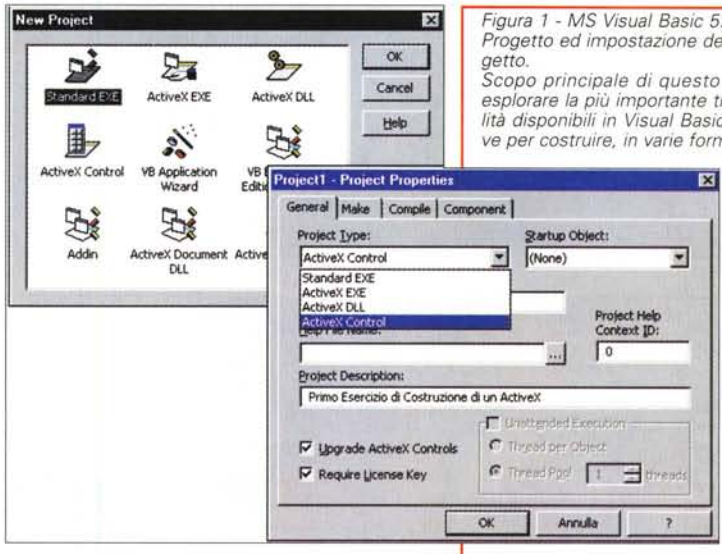

Figura 1 - MS Visual Basic 5.0 - Scelta del tipo di Progetto ed impostazione delle Proprietà del Progetto.
Scopo principale di questo articolo è quello di esplorare la più importante tra le nuove funzionalità disponibili in Visual Basic 5.0, quella che serve per costruire, in varie forme, componenti attivi riutilizzabili in vari ambiti. Con VB5 si possono realizzare eseguibili EXE, librerie DLL, documenti attivi, componenti ActiveX, che si concretizzano in un file con desinenza OCX. Molte delle attività inerenti la costruzione di tali componenti sono facilitate dalla presenza di efficaci Wizard. Facilita la costruzione dei componenti anche la possibilità di lavorare su più progetti, il primo per creare il componente ed il secondo per testarlo. Anche questa è una novità di Visual Basic 5.0. Qui vediamo la Box nella quale si sceglie il tipo di progetto che si vuole realizzare e quella nella quale si inseriscono le impostazioni "fini" del progetto in fase di sviluppo.

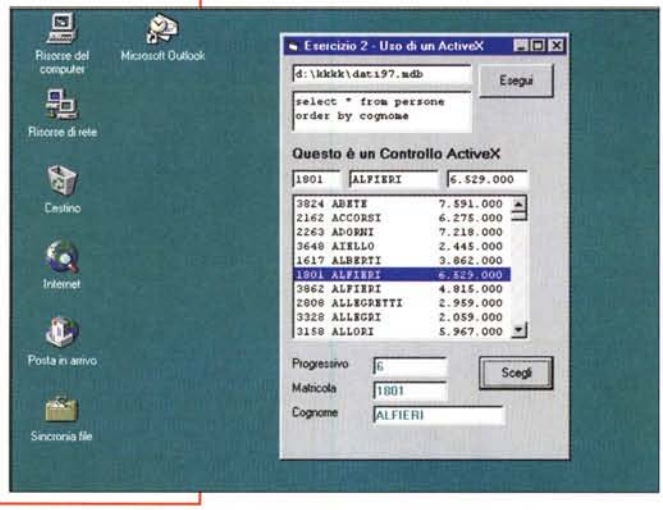

Figura 2 - MS Visual Basic 5.0 - Il nostro primo obiettivo: la pesca dei dati con il componente Lista3.
Vogliamo costruire un componente attivo (il cui nome sia LISTA3) che possa servire per "pescare" dati da un qualsiasi database, dati da utilizzare nelle più svariate situazioni. Cerchiamo di capirne il funzionamento osservando la figura: in alto: abbiamo due TextBox nelle quali digitiamo il nome del Database ed il Recordset desiderato (una Tabella o una Query, espressa in forma di comando SQL). Il pulsante Esegui imposta come Proprietà *Base e Dati del nostro nuovo oggetto il contenuto delle due box ed attiva un* Metodo, *il cui nome è Attiva, che scatena il riempimento della lista a tre colonne. In cima alla lista a tre colonne abbiamo piazzato tre altre TextBox (che fanno parte anch'esse del nostro componente) per vedere le informazioni del singolo elemento selezionato. Il pulsante Scegli trasferisce le tre nuove* Proprietà *(che si chiamano Progr, Matricola e Nominativo) dal controllo all'applicazione che lo sta usando.*

Anche Microsoft Internet Explorer è un HOST di ActiveX. L'attivazione avviene tramite il Tag HTML **<OBJECT>**. Tale Tag fa riferimento ad un controllo identificato tramite un **ID**. Se il controllo non è presente sul sistema occorre installarlo, ed in questo caso si utilizza la specifica **CODEBASE**, interna al Tag OBJECT, che in pratica permette di referenziare un ActiveX da un URL.
In questo caso il componente mancante viene scaricato, installato ed eseguito. I file scaricabili sono quelli con desinenza DLL, EXE e OCX, oppure CAB, che sono i file compressi autoinstallabili, ed INF, che sono i file contenenti le specifiche di installazione. I file vengono scaricati nella cartella **Windows\Occache** del sistema Client, e lì rimangono.
Su questo, che è forse l'aspetto più critico dell'attivazione di Internet, torneremo tra un po', dopo aver chiarito

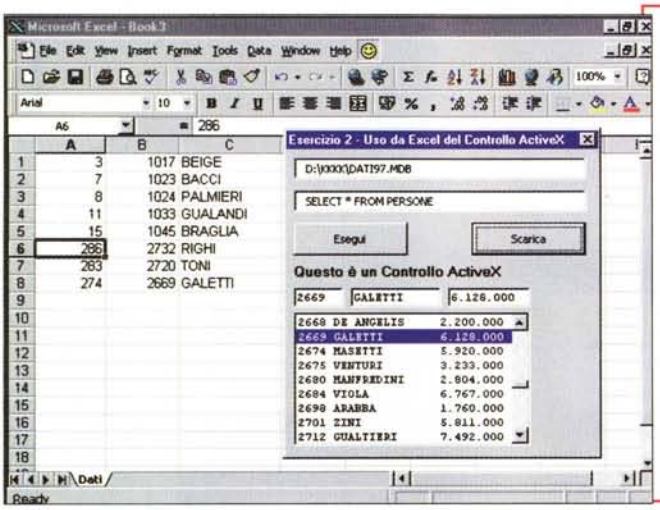

Figura 3 - MS Excel 97 - Utilizzo del controllo ActiveX Lista3.
Utilizziamo lo stesso componente ActiveX mostrato nella figura precedente questa volta in un'applicazione MS Excel 97. Nell'ambiente IDE di Microsoft Office 97 abbiamo realizzato una Form che contiene due caselle di testo, due pulsanti ed il nostro componente Lista3 (sinonimi di componente sono Controllo e ActiveX). La definizione del Database e del Recordset è analoga a quella utilizzata nell'esempio precedente. Quando si fa click sul pulsante Scarica vengono scaricate le tre proprietà, Progr, Matricola e Nominativo, direttamente sulle celle del foglio attivo. In pratica abbiamo realizzato un'applicazione che consente di scegliere i dati desiderati in maniera non preordinata. Se ci fosse una regola di selezione converrebbe costruire una Query.

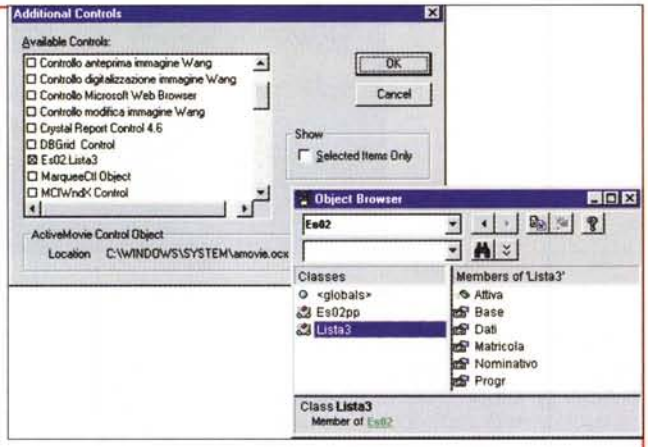

Figura 4 - MS Excel 97 - Utilizzo del controllo ActiveX Lista3.
Una volta generato ed installato, il controllo OCX diventa patrimonio comune dell'intero ambiente Windows. Il modo più semplice per utilizzarlo è quello di "pescarlo" dalla finestra Additional Controls, *presente in qualsiasi prodotto programmabile "ad oggetti", ed il metodo più semplice per programmarlo è quello di aiutarsi con la finestra* Object Browser *(Visualizzatore di Oggetti) che mostra, puntualmente, proprietà e metodi del componente selezionato. Da notare, in quest'ultima Box, la simbologia che distingue le Proprietà dai Metodi.*

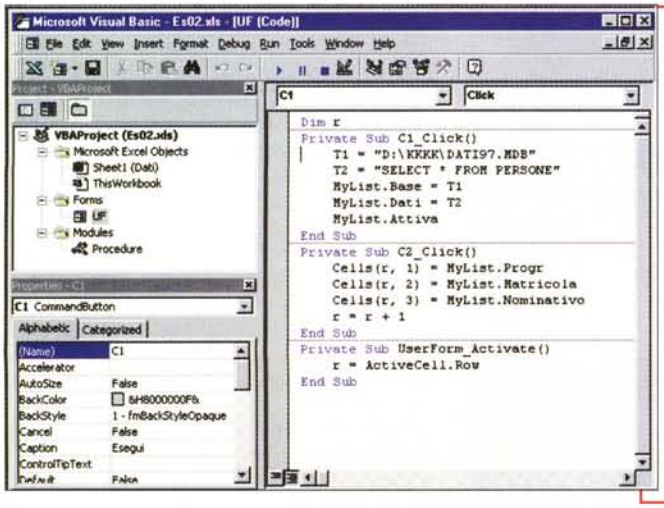

Figura 5 - MS Excel 97 - Utilizzo del controllo ActiveX Lista3: il listato VBA.
Analizziamo ora il codice VBA scritto per Excel. La situazione è mostrata chiaramente dalla ProjectView, *in cui notiamo la presenza degli oggetti* Sheet *e* ThisWorkbook, *di Excel, della* Form *UF (una UserForm) e di un Module che potrebbe contenere le varie procedure. In realtà tutto il codice è inglobato nella Form ed in particolare le due uniche routine sono associate agli eventi Click sui due Pulsanti. Quella sul primo pulsante imposta le proprietà Base e Dati dell'oggetto Lista3 (che nella nostra applicazione abbiamo chiamato MyList, avendo impostato la proprietà Name del componente) ed esegue il metodo Attiva. Il Click sul secondo pulsante trasferisce sulle celle del foglio i valori delle proprietà Progr, Matricola e Nominativo. La variabile R serve per gestire la riga corrente sulla quale trascrivere i dati scelti.*

cosa si intende per attivazione di Internet.

Cosa si intende per attivazione di Internet

Cercando di semplificare un discorso che oggettivamente è un po' complesso, in quanto fa riferimento all'architettura ed anche alla storia di Windows, possiamo affermare che ActiveX è un insieme di tecnologie che permettono di inserire un contenuto interattivo nelle pagine WEB. Le tecnologie di base sono tre.
La prima è costituita dagli **ActiveX Controls**, che sono i successori dei VBX e degli OCX, di cui abbiamo appena parlato e che costituiscono l'argomento principale del nostro articolo.
La seconda tecnologia consiste negli **ActiveX Document**. Si tratta di particolari estensioni che rendono normali documenti compositi (per intenderci quelli che si realizzano con i prodotti Office e che contengono oggetti OLE realizzati con altri prodotti OLE Server) di essere attivati interamente, come un oggetto unico, dall'interno di un ambiente Client.
Il primo esempio di tecnologia **ActiveX Document** è costituito dal **Raccoglitore di Office**, che non produce file propri ma semplicemente raccoglie ed assembla documenti realizzati con gli altri componenti della suite.
Il secondo esempio è costituito proprio da **MS Internet Explorer 3.0** che è un contenitore di documenti attivi. Come noto può visualizzare sia documenti HTML, sia documenti DOC, scritti con MS Word ed attivi, sia qualsiasi documento realizzato con qualsiasi prodotto, ma dichiarato attivo.
Il terzo esempio sarà

costituito, come già noto a molti di voi, dal **Desktop** della prossima versione di Windows 95 (anticipata in **MS Internet Explorer 4.0)** che rende lo stesso desktop un contenitore di documenti attivi.
La terza tecnologia, **ActiveX Scripting**, definisce le caratteristiche dei prodotti Host, quelli che sono in grado di eseguire programmi e documenti attivi. Ad esempio MS Internet Explorer è un Host ActiveX Scripting e dispone di un motore per eseguire script sia in linguaggio **VB-Script** che **JScript**.
Rimanendo strettamente in ambito Microsoft, non si può non notare l'omogeinizzazione dei vari sottinsiemi del linguaggio VB, con VB5, VBA e VB-Script, e soprattutto l'omogeinizzazione dei prodotti che sono tutti in grado di condividere i vari componenti.
Torniamo all'argomento principale del nostro articolo, la sviluppo di componenti ActiveX con VB5 e il loro utilizzo con altri prodotti ed in altre situazioni.

Figura 6 - www.activex.com - Il sito Internet dedicato agli ActiveX. La Tecnologia ActiveX è una tecnologia trasversale, in quanto riguarda tutti i prodotti di sviluppo che operano all'interno dell'ambiente Windows, per cui il suo approfondimento può partire da ciascuno di questi prodotti. Noi partiremo dal Visual Basic 5.0 che, come detto, per la prima volta permette di costruire componenti di questo tipo. L'argomento ActiveX, pur essendo abbastanza nuovo, è molto documentato. Viene ben trattato nei manuali del Visual Basic 5 ed esistono, nel catalogo Microsoft Press, numerosi testi su tale argomento. Su Internet sono dedicate ad ActiveX molte pagine del Sito Microsoft e molti siti riservati agli sviluppatori.

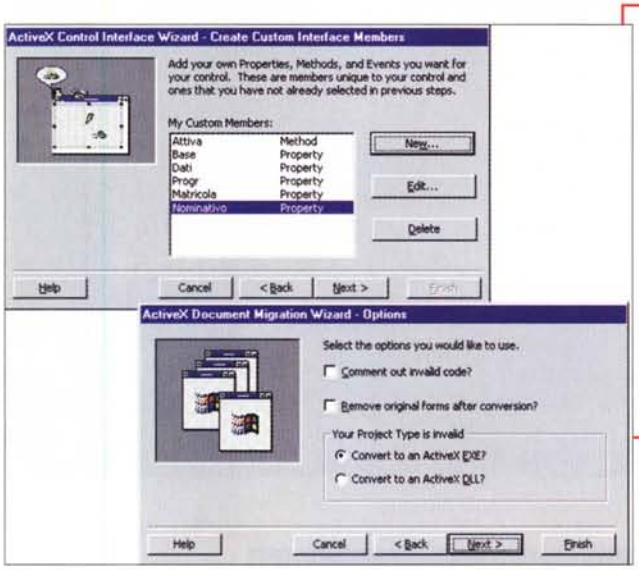

Figura 7 - MS Visual Basic 5.0 - Immagini dai Wizard. Riguardano la creazione dei componenti attivi due Wizard, quello che guida nella costruzione di un ActiveX e quello che converte un'applicazione VB5 in un documento attivo. Si tratta ovviamente di procedure molto comode, ma che vanno usate solo da chi sa esattamente quale sia l'obiettivo da raggiungere e che magari lo saprebbe raggiungere anche senza usare il Wizard.

L'ambiente di sviluppo VB e la scelta del tipo di attività

Nella prima figura dell'articolo abbiamo inserito due Box, quella che appare quando si esegue il comando File New Project, e che presenta una decina di "varianti", e quella che mostra le proprietà del progetto in cui si controlla, o al limite si imposta, il tipo di progetto da sviluppare. E' chiaro infatti che il fatto che si crei un eseguibile EXE, oppure una libreria DLL, oppure ancora un controllo ActiveX, non può che essere una scelta iniziale, da fare quando si inizia un nuovo progetto.
Il nostro obiettivo è quindi quello di realizzare un ActiveX, che si concretizza, come detto, in un file OCX, riutilizzabile in altri programmi.
Partiamo dalla fine vedendo subito come funziona il controllo ActiveX che vogliamo costruire. Lo vediamo, in figura 2, inserito in un'applicazione VB, ed, in figura 3, inserito in un'applicazione Excel. Il nome del controllo è Lista3 ed è presente nel file ES02.OCX.
Il nostro controllo serve per pescare, in maniera non preordinata, dati da una Tabella di un Database. I dati pescati, o meglio scelti uno ad uno, devono essere resi disponibili nell'applicazione che ospita il controllo (VB 5.0 ed Excel 97).
Il funzionamento dell'oggetto, che internamente usa comandi DAO, è descritto nella didascalia di figura 2.
E' chiaro che una volta realizzato l'oggetto (oppure una volta disponibile, se realizzato da altri) è facilmente utilizzabile, in quanto l'ambiente Host dispone delle classiche Box (le vediamo in figura 4) con le quali si scelgono i controlli, tra quelli presenti sul sistema, e si gestiscono **Proprietà** e **Metodi** del controllo. Le Box sono la **Insert Controls** e la **Object Browser.**
In figura 5 vediamo il codice del programma, scritto in VB5 per Excel 97, che permette di usare il controllo. Anche in questo caso per la descrizione del listato vi rimandiamo alla figura e alla sua didascalia.

Come documentarsi

Mentre, come abbiamo appena visto, è relativamente facile usare un controllo ActiveX, in quando esistono

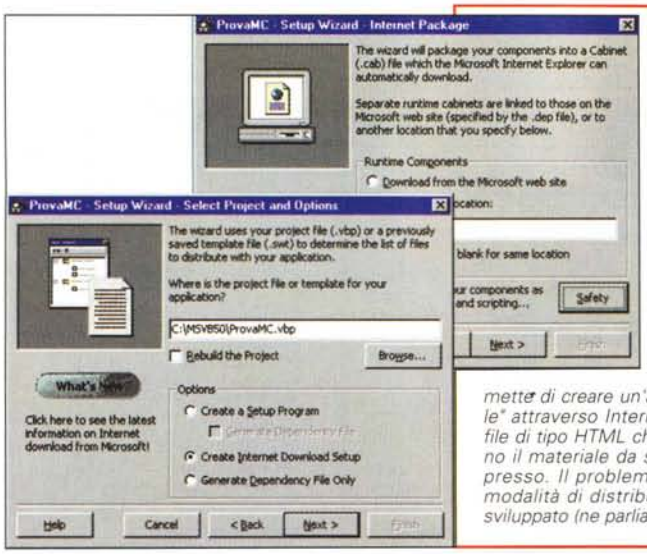

Figura 8 - MS Visual Basic 5.0 - Il Setup Wizard. I vari Wizard di cui abbiamo parlato nelle precedenti didascalie (e anche altri di cui non abbiamo parlato in questo articolo) si lanciano dall'interno dell'ambiente di sviluppo di Visual Basic. C'è anche il Setup Wizard, che invece è esterno, il cui compito è quello di produrre il file eseguibile "finale". Una variante di questa procedura (la vediamo nella figura) è quella che permette di creare un'applicazione "downloadabile" attraverso Internet. Vengono generati sia file di tipo HTML che file CAB, che contengono il materiale da scaricare in formato compresso. Il problema successivo riguarda le modalità di distribuzione del materiale così sviluppato (ne parliamo nel testo).

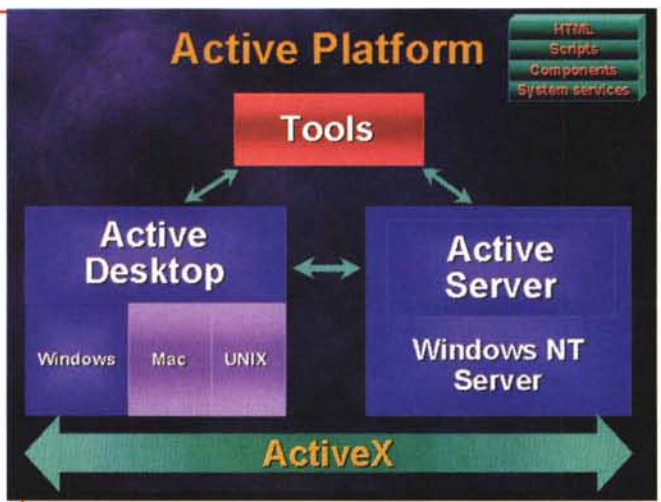

Figura 9 - Dall'Active Desktop all' Active Server.
L'attuale tecnologia, e gli esercizi che vi proponiamo nell'articolo, riguardano la possibilità di utilizzare componenti che debbono essere distribuiti su ciascun Client. In un prossimo futuro, in termini di prodotto parliamo di Windows NT Server 5.0, l'orientamento sarà quello di far lavorare di più il Server e meno i Desktop. Il fatto, ad esempio, che i componenti ActiveX potranno esser lasciati sul Server e da lì richiamati all'occorrenza ne faciliterà la gestione e la manutenzione.

strumenti che ne aiutano la gestione, non è altrettanto facile sia progettarli che costruirli.
Nel primo caso la difficoltà sta nel decidere quale parte di attività vada assegnata al controllo, e quale lasciata al programma host, in modo che poi convenga effettivamente riutilizzare il controllo anziché scrivere un nuovo codice.
Nel secondo caso la difficoltà sta nel dover pensare in termini di Proprietà del controllo, in termini di Metodi subiti dal controllo, in termini di Eventi che si scatenano sul controllo. In prima approssimazione si può affermare che le Proprietà del controllo equivalgono a Variabili che si possono impostare oppure leggere, i Metodi equivalgono a Procedure che vengono eseguite. Per gli Eventi è un po' più difficile trovare un parallelismo.
Insomma il controllo ActiveX va "pensato" in termini di Proprietà, Metodi ed Eventi e qui sta la difficoltà, perlomeno inziale, per chi non è abituato a lavorare per componenti.
Per quanto riguarda la documentazione sono fondamentali i manuali del Visual Basic 5.0 (che descriviamo nella prova pubblicata in questo stesso numero di MC), i tanti siti Internet dedicati a tale tecnologia (figura 6), i testi MS Press o di altre case editrici tecniche.
Anche i **Wizard** (ne vediamo esempi nelle figure 7 ed 8), che aiutano il programmatore nelle fasi più onerose del lavoro, possono servire a capire i meccanismi su cui si basa la Programmazione Object Based.
Infine, prima di vedere come si fa a costruire un ActiveX (dopo aver visto come si fa ad usarlo), ricordiamo che la tecnologia ActiveX è ancora agli inizi. Diventerà ancora più importante, specialmente in un'ottica aziendale, dove le applicazioni sono "Mission Critical", quando sarà attivo il Server. Sarà, in altre parole, solo il Server a contenere, in copia unica, protetta, sicura, i componenti,

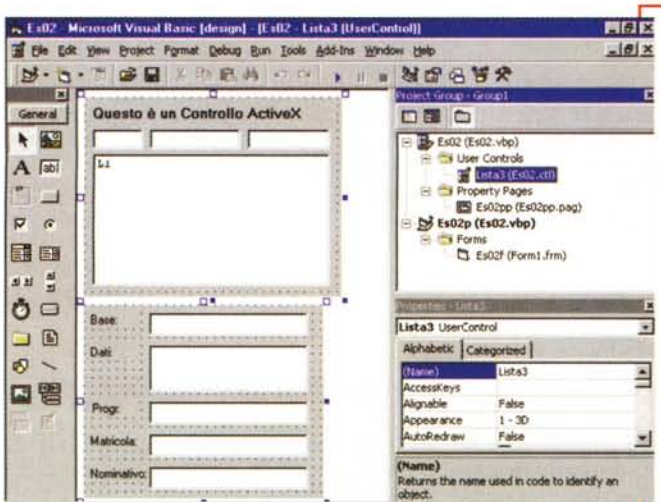

Figura 10 - MS Visual Basic 5.0 - L'applicazione che crea l'ActiveX.
Dopo aver visto come si usa, lavorando con Excel, il nostro ActiveX, vediamo come si costruisce, lavorando con Visual Basic 5.0. Nel suo ambiente di sviluppo vediamo in alto a destra la Project Window, che contiene il progetto (il primo dei due, che si chiama ES02) che a sua volta comprende due Form, quella che coincide con la parte esteriore del nostro oggetto, e quella che ne mostra le proprietà "Custom". Questa seconda form, che è generabile con uno specifico Wizard, è quella che appare quando si apre la Box con le proprietà personalizzate del nuovo controllo. Secondo la terminologia vigente in VB5, si tratta di una User Form e di una Properties Page Form. Il codice, che riportiamo nel testo, comprende sia parti tradizionali, ad esempio quelle che fanno funzionare la Form, sia quelle che servono per gestire Proprietà e Metodi.

```
Dim DD As Database              ' dichiarazioni
Dim RR As Recordset
Dim BV, DV As String

Private Sub L1_Click() ' trasferimento dati dalla Lista nelle TextBox
 D1 = Left(L1, 4)
 D2 = Mid(L1, 6, 12)
 D3 = RTrim(Mid(L1, 21, 15))
End Sub
```

listato A

```
Private Sub UserControl_Initialize()  ' evento iniziale del controllo
  BV = ""
  DV = ""
End Sub

Private Sub UserControl_Terminate()   ' evento finale del controllo
    ' non utilizzato
End Sub
```

listato B

```
Public Function Attiva()                                    '   definizione
del metodo ATTIVA
  Set DD = OpenDatabase(bv)                                 ' apertura del
DB
   Set RR = dd.OpenRecordset(dv,dbOpenSnapshot)   ' creazione del Re-
cordSet
  L1.Clear                                                  '   svuotamento
della Lista
  Do While Not RR.EOF                           ' scorrimento del Re-
cordSet
     KK = RR!cod & " "                          ' lettura dei campi di
interesse
     KK = KK & Left(RR!cognome + Space(10), 12)
     KK = KK & Right(Space(10) + Format(RR!importo, "#,###"), 12)
     L1.AddItem KK                                          '   riempimento
della Lista
     RR.MoveNext                                            ' record succes-
sivo
  Loop                                                      ' fine scorri-
mento
  L1.ListIndex = 0
End Function
```

listato C

anche se i Client che li utilizzeranno saranno migliaia (figura 9 per promemoria).

Andiamo un po' più a fondo

In figura 10 vediamo il nostro componente in lavorazione. Comprende due form, una **User Form**, che è la parte esteriore dell'applicazione e la **Property Form**, che può essere generata da un Wizard e che contiene le varie proprietà, leggibili ed impostabili, del nuovo controllo. La Property Form corrisponde alla Box Custom Property del controllo.

La User Form è invece una Form normale, contiene componenti normali e parti di codice normale. Nel nostro caso tutto ciò che serve per attivare il Database, leggere la Tabella, riempire la Lista, trasferire i dati dalla Lista alle Text Box. Riportiamo le parti di codice "normale"(vedi listato A):

C'è poi una parte di codice specifica del controllo ActiveX. Occorre infatti definire le sue **Proprietà**, che in pratica sono Variabili che vanno impostate, lette, e sulle quali o dalle quali, spostare valori, ed occorre definire **Metodi**, che sono procedure, o funzioni che "movimentano" l'oggetto.

Il controllo dispone di due Eventi specifici, utili per le eventuali inizializzazioni da eseguire quando si attiva il controllo e per le eventuali riconfigurazioni finali(vedi listato B).

Il Metodo **ATTIVA** può essere una Function oppure una Sub. La sua programmazione è standard. In pratica è semplicemente il fatto che stiamo lavorando su un ActiveX e non su un'applicazione standard che traduce la Function in un Metodo (vedi listato C).

Qui vediamo invece la definizione delle proprietà. E' fondamentale capire la differenza tra le proprietà di scrittura, quelle che servono per trasferire i dati verso il controllo, e quelle di lettura, che servono, al contrario, per riportare dati dal controllo all'applicazione. Nel nostro caso quelle di scrittura sono Base e Dati (l'istruzione è **Let**), e servono per riportare sull'ActiveX Database e RecordSet di interesse. Quelle di lettura (istruzione **Get**) servono per leggere i campi (ricordiamo che abbiamo usato una Lista semplice, nella quale abbiamo simulato la creazione di tre colonne) della riga scelta nella Lista (vedi listato D).

Ci fermiamo qui. Non ci interessa-

```
Public Property Let Base(ByVal vNewValue As Variant)
    BV = vNewValue
End Property

Public Property Let Dati(ByVal vNewValue As Variant)
    DV = vNewValue
End Property

Public Property Get Progr() As Variant
    Progr = L1.ListIndex + 1
End Property

Public Property Get Matricola() As Variant
    Matricola = Left(L1, 4)
End Property

Public Property Get Nominativo() As Variant
    Nominativo = Trim(Mid(L1, 6, 12))
End Property
```

listato D

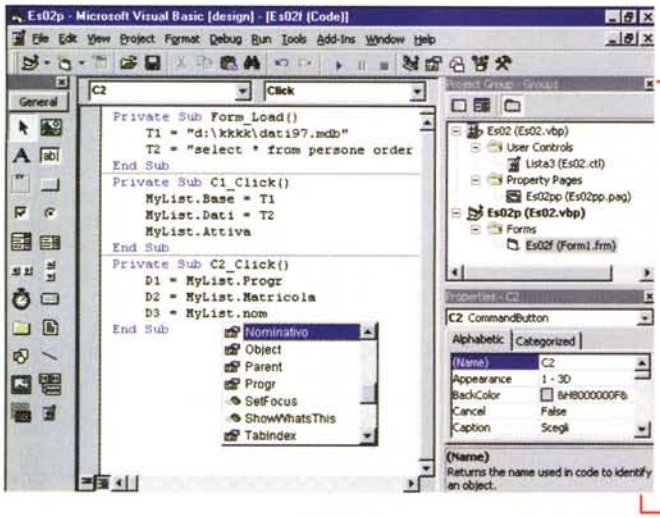

Figura 11 - MS Visual Basic 5.0 - L'applicazione che usa l'ActiveX. Il risultato dell'esecuzione di questo programma lo abbiamo visto in figura 2. Poiché siamo in fase di test vediamo, nell'ambiente IDE di VB5, ambedue i due progetti, quello che crea e quello che testa il nuovo controllo OCX. Questa possibilità, anche questa è una novità del Visual Basic 5.0, è molto comoda proprio quando si sviluppano componenti che vanno via via provati. Notiamo anche la semplicità del codice necessario per usare il componente ed il fatto che lo stesso codice sia del tutto analogo a quello scritto nella Macro Excel.

va costruire un controllo complesso, ma solo verificare quanto il fatto di costruire un componente riutilizzabile complichi il nostro lavoro con VB.
In definitiva lo sforzo in più è pressoché irrilevante, specie quando il componente (come nel nostro caso) assomiglia molto ad una comune sottoprocedura.
Anche da un punto di vista concettuale lo sforzo è ridotto, quando si siano ben capite le equazioni **Metodo=Procedura** e **Proprietà=Variabile**.
Degli **Eventi**, che non abbiamo toccato, e che probabilmente ci complicheranno la vita, parleremo un'altra volta.
Va infine detto che buona parte del lavoro si può far svolgere ai numerosi Wizard, che, ad esempio, possono aiutarci nella creazione delle varie Proprietà e dei vari Metodi.

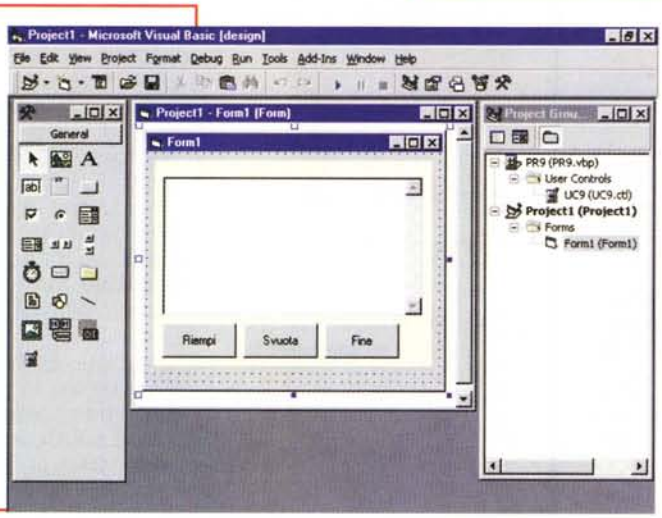

Figura 12 - MS Visual Basic 5.0 - Costruzione di un semplice componente ActiveX. Nome: PR9.
Ci trasferiamo armi e bagagli su Internet. Il nostro obiettivo è quello di realizzare un'applicazione VB5, semplicissima che più semplice non si può, salvarla come componente ActiveX, provarla in un'altra applicazione VB (l'istantanea immortala questo momento) e poi utilizzarla all'interno di una normale pagina HTML. Un click sul pulsante Riempi riempie la TextBox con la frase "Prova MC", mentre un click sul pulsante Svuota la ripulisce.

Ed ora attiviamo Internet

Dopo aver visto come sia possibile, con VB5, sviluppare un controllo ActiveX e come sia facile usarlo in un'applicazione scritta con Visual Basic stesso oppure con Excel (anch'esso Host di ActiveX), passiamo a parlare di Internet ovvero di come sia possibile sviluppare qualcosa, un componente oppure un'intera applicazione, che renda attiva una comune pagina HTML.
Per prima cosa realizziamo con VB un programma semplicissimo (lo vediamo in figura 12). Questo consiste in una Form con una TextBox e tre pulsanti. Il click sul primo riempie la TextBox con la scritta "Prova MC", il click sul secondo pulisce la Text e il terzo fa terminare l'applicazione.
Se si dichiara subito che si tratta di un ActiveX Control si lavora direttamente su una User Form. Si inseriscono oggetti, si scrivono routine, come in un programma normale.
In questo caso non vogliamo definire Proprietà, Metodi, Eventi, vogliamo solo far funzionare il codice sottostante i pulsanti.

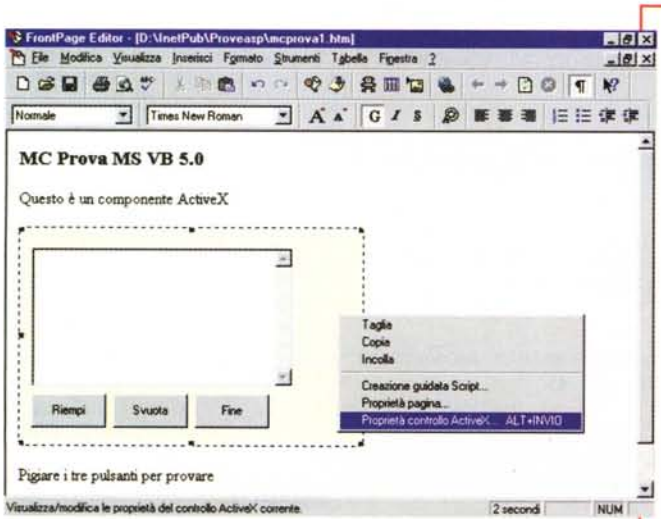

Figura 13 - MS Front Page 97 - Creazione di una pagina HTML che contiene il componente PR9.
Per creare la pagina HTML "attiva" usiamo l'Editor di MS Front Page, utile non fosse altro che per la funzionalità che permette di inserire controlli ActiveX. Notiamo anche come, una volta inserito il nostro controllo, sia attivo il tasto destro del mouse, che apre il Quick Menu che permette di vedere le proprietà dell'oggetto. Purtroppo non c'è uno strumento tipo Object Browser e l'eventuale programmazione del controllo va fatta "a manina".

Chiamiamo il progetto **PR9.VBP** e la Form, che costituisce il controllo, la chiamiamo **UC9.CTL**. Per provare il funzionamento del controllo ActiveX basta chiudere l'User Form e creare un nuovo progetto, normale, nella cui Form possiamo inserire il nostro ActiveX in lavorazione, che è già disponibile nella Toolbox.
Lanciando il secondo progetto si può verificare il funzionamento del controllo, il quale dispone comunque di una serie di proprietà standard.
E' molto importante, lo abbiamo già detto, il fatto che si possa lavorare contemporaneamente sui due progetti e che VB permetta di salvarli ambedue (con Form, Modules, ecc.) in un unico Project Group.
Quando tutto funziona basta eseguire il comando File Make OCX per produrre il file con il controllo.
Ora dobbiamo usare il nostro PR9.OCX in un file HTML.
Apriamo l'Editor di **MS FrontPage** con il quale creiamo una semplice paginetta in cui, sfruttando il comando Inserisci/Altri Componenti/Controllo ActiveX, piazziamo il nostro controllo. Il nostro controllo non prevede Eventi, non subisce Metodi, non ha Proprietà particolari, oltre a quelle standard.
Se andiamo a curiosare tra le proprietà standard ne troviamo due molto interessanti, che ritroveremo in seguito: la **ID** che identifica il nostro componente nell'applicazione che lo ospita e la **CodeBase** che invece identifica il file OCX (vediamo l'Editor di FrontPage in figura 13).
Non ci resta da fare altro che provare il funzionamento dell'HTML generato dall''Editor di FP.
In figura 14 vediamo MS Internet Explorer con la nostra paginetta ed in una finestrella vediamo l'HTML relativo.
Tutto ruota attorno al Tag **OBJECT**, che contiene le specifiche **ID** e **CodeBase**, che servono per identificare l'oggetto, nel caso ad esempio attivasse un evento, e per identificare da dove scaricare il componente, nel caso il sistema Client non ne disponesse.
E' chiaro che è proprio questo l'aspetto più importante, in quanto riguarda la distribuzione dell'OCX, la sicurezza del Client, ecc. Per ora ci accontentiamo del fatto che la nostra applicazione funziona su Internet. In successivi articoli approfondiremo il problema della distribuzione "sicura" del materiale.
Per finire vi mostriamo come sia facile prendere un'applicazione Visual Basic e trasformarla in un'applicazione Internet (in figura 15 vediamo l'applicazione ed, in una piccola finestrella, il suo listato VB originario). Della trasformazione, in questo caso si parte da un programma VB e si ottiene un Documento Attivo, se ne occupa un Wizard.
E' chiaro che non può essere tutto così semplice.
Ad esempio nel primo caso (PR9) il pulsante che esegue la fine del programma (istruzione VB Unload Me) non funziona se usato in un ActiveX.
Nel secondo caso perdiamo del tutto il concetto di Resize, che esiste in una Form VB, ma non esiste in una pagina HTML.
In sostanza per sfruttare bene le enormi potenzialità di VB5 ed in particolare le sue interessanti sinergie con Internet bisogna conoscere molto bene tutti e due i... mondi.

MC

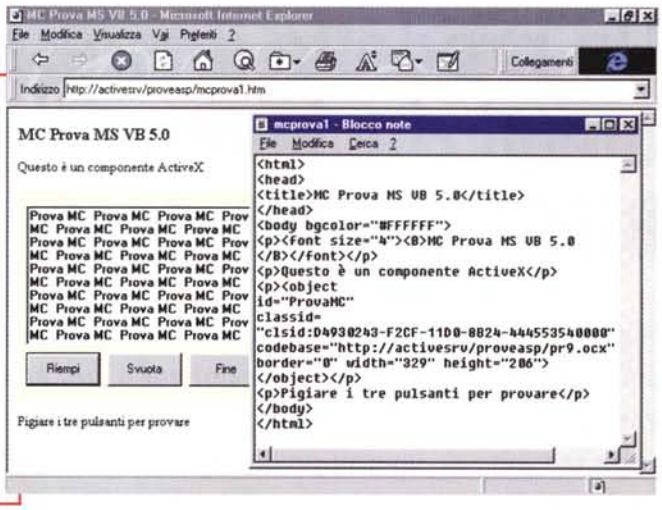

Figura 14 - MS Internet Explorer 3.0x - Utilizzo, in una pagina HTML, del componente PR9. Salvato il file MCProva.HTM lo possiamo vedere con Internet Explorer. Possiamo anche utilizzare il controllo ActiveX agendo sui due pulsanti che, in pratica, rendono attiva la pagina. Il terzo pulsante invece produce un errore. Contiene l'istruzione UNLOAD che funziona in un'applicazione VB, ma non in una pagina HTML.

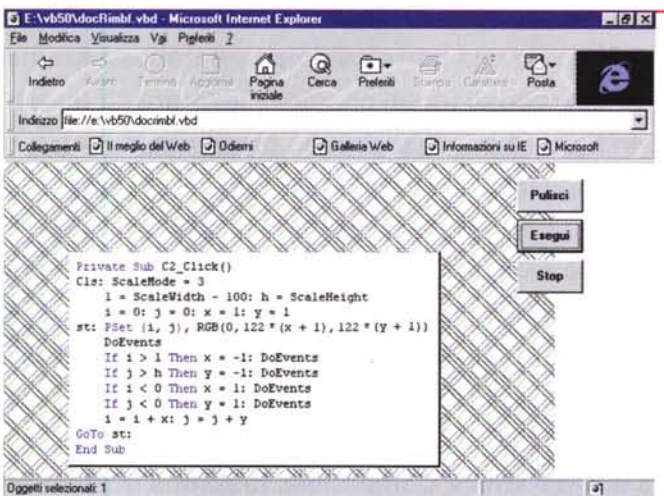

Figura 15 - MS Visual Basic 5.0 - Esasperazione del concetto di Active Document. Abbiamo realizzato un programmino con Visual Basic, del quale potete leggere il listato nella finestrella in mezzo alla videata, e che consiste in una Form con tre Pulsanti. Al click sul secondo dei tre, viene eseguito il codice che genera una linea che rimbalza sui quattro bordi della Form, cambiando via via colore. L'unità di misura della Form è il Pixel (proprietà ScaleMode=3) e nel listato c'è l'istruzione DoEvents che permette di intercettare il click sugli altri due pulsanti Pulisci e Stop. Lanciando il Wizard ActiveX Document Migration viene generato, gratis, un documento attivo immediatamente utilizzabile all'interno di un contenitore di documenti attivi, come il Binder di Office o il Microsoft Explorer. Estremizzando, e tacendo alcune considerazioni che comunque dovremo fare, possiamo affermare che un'applicazione VB è potenzialmente anche un'applicazione Attiva per Internet.