Programmazione ad Oggetti: Metodi e Proprietà

E' bene ricordare che il VBA è un linguaggio di programmazione *orientato agli oggetti* che consente di gestire le applicazioni con semplicità e *la Programmazione ad Oggetti* è un sistema di organizzazione del codice di un programma mediante raggruppamento all'interno di oggetti, ovvero singoli elementi che presentano *Proprietà* e *Metodi*, in pratica non è più il programma a gestire il flusso di istruzioni, ma sono gli oggetti che interagiscono tra di loro intervenendo sulle azioni del programma, introducendo il concetto di Evento che tratteremo nella prossima lezione. Per fare un esempio concreto, possiamo dire che tutto il mondo intorno a noi è pieno di oggetti: la tastiera, il mouse, lo schermo e così via, quello che contraddistingue un oggetto è una serie di caratteristiche o Proprietà, infatti il mouse è dotato di una rotellina e di due tasti, un oggetto, oltre alle proprietà, possiede anche dei Metodi ossia delle funzioni attraverso le quali esercitare delle azioni.

Nell'esempio appena citato, (quello del mouse) tenendo premuto il tasto sinistro e facendo scorrere il mouse si seleziona una o più parole di una riga di testo. Riferendoci alla nostra cartella di lavoro possiamo dire che essa è costituita da oggetti di varia natura, come gli oggetti grafici o i controlli (finestre di dialogo, pulsanti ecc.) incollati su un foglio, ma non è questa la peculiarità di un foglio elettronico la cui ossatura è costituita, partendo dal livello più basso, da :

- celle
- intervalli
- fogli
- cartelle di lavoro

Se osserviamo questo insieme di oggetti come un albero genealogico il capostipite, o il culmine, spetta all'oggetto, denominato *Application*, che rappresenta, tutti gli Oggetti di Excel. Si tratta appunto di Excel stesso. Proseguendo nella discesa genealogica troveremo i seguenti oggetti:

- L'oggetto Workbook : Che è la cartella di lavoro (cioè il nostro file)
- L'oggetto Worksheet : Che è il foglio di lavoro (Foglio1, Foglio2 ecc...)
- L'oggetto Range : Che è un intervallo di celle (A1: B12, C1:D12, ecc...)

Sintatticamente possiamo affermare che ciascun oggetto fa parte di una famiglia o classe e l'accesso al singolo membro di ciascuna classe si effettua attraverso metodi, *pluralistici*, cioè da una pluralità di metodi che collaborano e interagiscono con i vari oggetti in maniera omogenea e ai quali corrispondono insiemi di oggetti quali :*Workbooks*, *Worksheet*, *Range* e *Cells*, inoltre i membri più elevati si possono omettere nel caso che il soggetto sia attivo; vedremo meglio questo passaggio fra poche righe.

Fin qui abbiamo delineato i componenti principali cercando di esporre come vengono interpretati dal Visual Basic applicato al foglio elettronico, ma l'obiettivo vero è quello di focalizzare gli oggetti che formano l'ossatura, il nucleo di uno *spreadsheet* (foglio di lavoro), al cui centro, vi sono intervalli e celle con il loro contenuto di dati da elaborare o formule e il loro inquadramento nel mondo Visual Basic è fondamentale e aiuterà a capire meglio tutto il resto, vediamo di interpretare quanto appena affermato usando il Visual Basic. Con queste sintassi

Workbooks("Lezione2.xls") e *Worksheets("Foglio1")*

Si individuano rispettivamente la cartella e il foglio di lavoro (notare i loro nomi virgolettati dentro le parentesi), ma dobbiamo però tenere presente che un elemento può anche venire individuato tramite un indice, il quale può essere o il numero di ordine o il nome fra virgolette, per capire meglio il concetto di

indice possiamo dire che la sintassi Workbooks(2) e Workbooks("Lezione2.xls") puntano entrambe alla stessa cartella Lezione2.xls a patto che questa sia la **seconda** fra quelle aperte contemporaneamente da Excel.

Infatti se abbiamo solo la cartella Lezione2.xls aperta, la sintassi esatta diventa *Workbooks(1)*. A questo punto è abbastanza chiaro che l'indice che usiamo fra parentesi nell'oggetto Workbooks varia ed è strettamente legato al numero di cartelle aperte nel momento dell'esecuzione di questa istruzione, pertanto possiamo far notare che è possibile scrivere questa istruzione in tre diversi modi:

- *Application.Workbooks(1).Worksheets(1).Range("A1:B 10")*
- *Application.Workbooks("Lezione2.xls").Worksheets(1).Range("A1:B10")*
- *Application.Workbooks("Lezione2.xls").Worksheets(" Foglio1").Range("A1:B10")*

Ricordate che poco sopra abbiamo però affermato che i membri più elevati si possono omettere quando sono attivi, per cui:

- Se abbiamo Excel aperto
- Se la cartella Lezione2.xls è aperta
- Se infine ci troviamo nel Foglio1

Possiamo ridurre tutto il listato ad un semplice Range("A1:B10"), in caso contrario credo che sia abbastanza chiaro come agire usando gli indici per individuare il nostro intervallo (o Range). Si deve inoltre prestare attenzione alle proprietà "pluralistiche" che poco sopra abbiamo citato: le prime volte è facile scordarsi del plurale, scrivendo *Workbook*("Lezione2.xls") anziché Workbooks("Lezione2.xls") oppure *Worksheet*("Foglio1") invece diWorksheets("Foglio1").

Vediamo qualche esempio di codice da applicare a quanto esposto finora, ma sopratutto vediamo come automatizzare l'esecuzione di routine creando un pulsante, ad esempio apriamo la nostra cartella ("Lezione2.xls") e rientriamo nel Visual Basic Editor. Questa volta scriveremo noi una macro direttamente senza affidarci all'aiuto del Registratore di macro utilizzato nella precedente lezione. Ponendoci al di sotto dell'ultima riga della macro precedente (basta porre il cursore alla fine di End Sub e premere una o due volte Invio), scriviamo il seguente codice

```
Sub Nascondi_Foglio()
Worksheets(2).Visible = False
MsgBox "Il Foglio 2 è sparito"
End Sub
```

Commentiamo questo codice: La routine inizia con *Sub* (seguito dal nome della routine) e termina con *End Sub*. L'oggetto Worksheets(2) ossia il Foglio 2, è seguito da una sua proprietà che si evidenzia con *.visible*, si richiama perciò la proprietà *visible* che consente al foglio di essere visibile o meno a seconda del fatto che tale proprietà sia dichiarata *vera* [True] o *falsa* [False] (nel nostro caso è stata dichiarata uguale a False per cui indica che il foglio sarà nascosto).

MsgBox "Il Foglio 2 è sparito" è invece una funzione di VBA che permette di mostrare all'utente un messaggio che nella sua forma più semplice riporta un messaggio di avviso per l'utente (il testo racchiuso fra virgolette) ed un bottone di OK necessario per la sua chiusura. Torniamo ora ad Excel e facciamo click col destro del mouse sulla barra degli strumenti, ci comparirà un box con delle voci, scegliamo *Moduli*

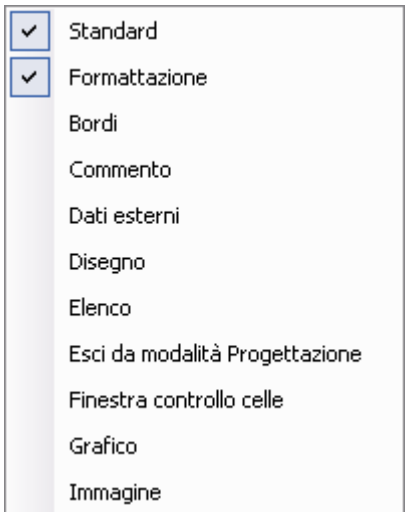

Fig. 1

Ci comparirà un nuova barra flottante che possiamo ancorare alle altre barre degli strumenti semplicemente trascinandocela.

Fig. 2

Premiamo sul pulsante evidenziato dalla freccia rossa e poi spostandoci sul foglio di lavoro teniamo premuto il pulsante di sinistra del mouse, trasciniamolo fino a raggiungere le dimensioni desiderate ed al rilascio ci comparirà il nostro pulsante. Contestualmente si aprirà anche la finestra per assegnare la macro (Nascondi_Foglio) al pulsante appena creato

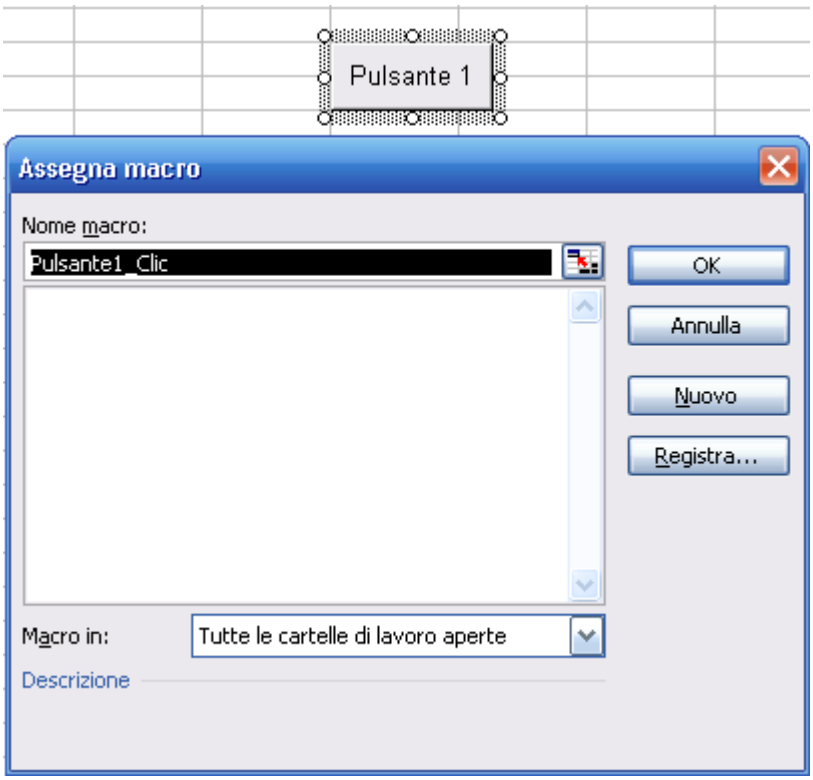

Fig. 3

Stessa operazione e identico risultato si può ottenere inserendo un'immagine sul foglio (si dovrà attivare la barra degli strumenti *Disegno*) e quindi tramite il solito menu contestuale ottenuto cliccando col destro sull'immagine, assegnargli la macro. Possiamo ridimensionare il pulsante, editarne il testo, cambiarne i caratteri etc. facendo clic sopra al pulsante stesso col pulsante destro del mouse e scegliere, tra le voci del menù contestuale che compare, quella di cui abbiamo bisogno. Se invece vogliamo spostare il pulsante, facciamo ancora clic col destro del mouse, sparirà il menù contestuale e rimarrà evidenziato il pulsante

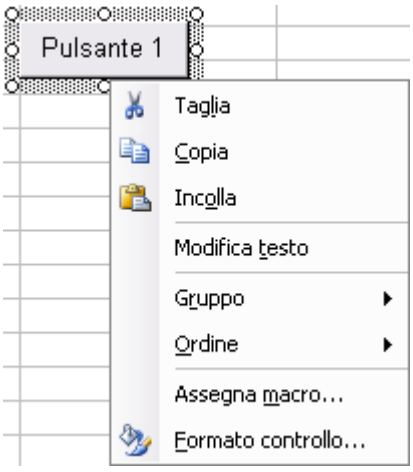

Fig. 4

Possiamo allora spostarlo a piacimento e ridimensionarlo trascinandolo negli angoli

Fig. 5

Per versioni di Excel superiori alla 2003 *si deve seguire il percorso:* ***File - Opzioni - Personalizzazione barra multifunzione*** *e nel box Personalizza barra Multifunzione mettere il flag al campo Sviluppo, come da figura sottostante*

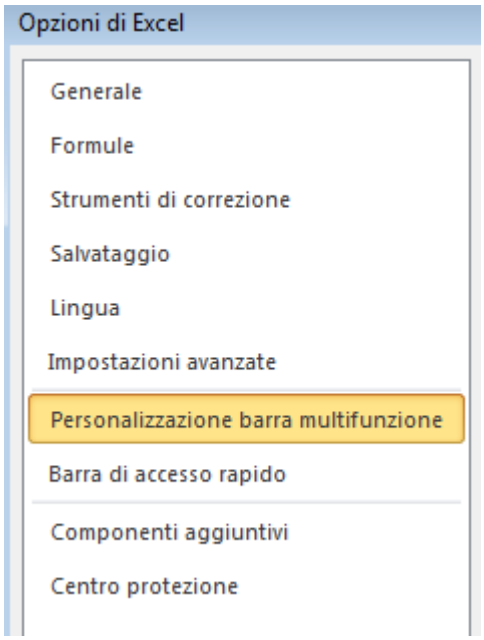

Fig. 6

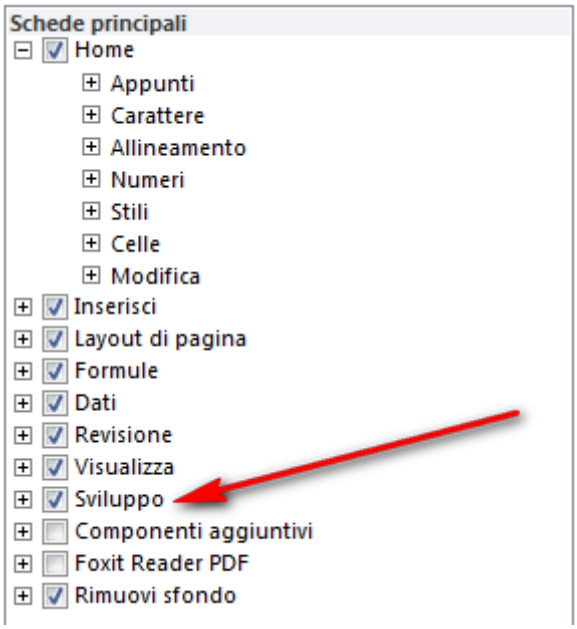

Fig. 7

A questo punto è comparso un altro menu nella barra multifunzione, il menu *Sviluppo*, tramite il sotto menu *Inserisci* scegliere il pulsante e procedere come spiegato poco sopra per la Ver. 2003

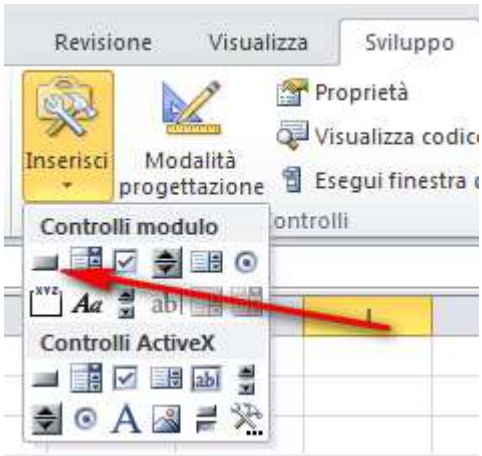

Fig. 8

PS: E' possibile usare una forma geometrica invece di un pulsante a cui assegnare una macro, in questo caso si procede in questo modo: dal menu **Inserisci - Forme**

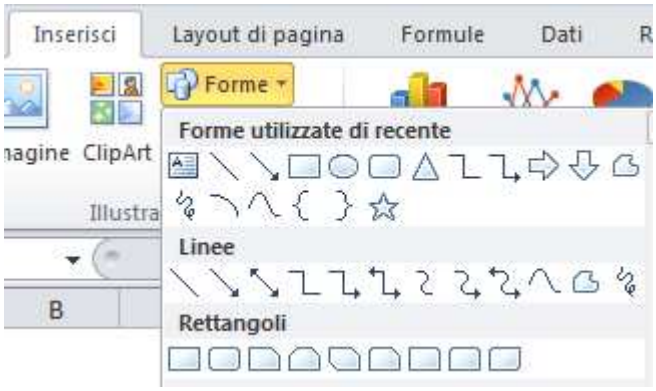

Fig. 9

Scegliere il rettangolo e disegnarlo sul foglio, a operazione compiuta verrà riportata in automatico la finestra delle macro, in alternativa è possibile farla comparire cliccando col tasto destro del mouse sulla forma e nel menù che compare scegliere la voce *Assegna Macro*

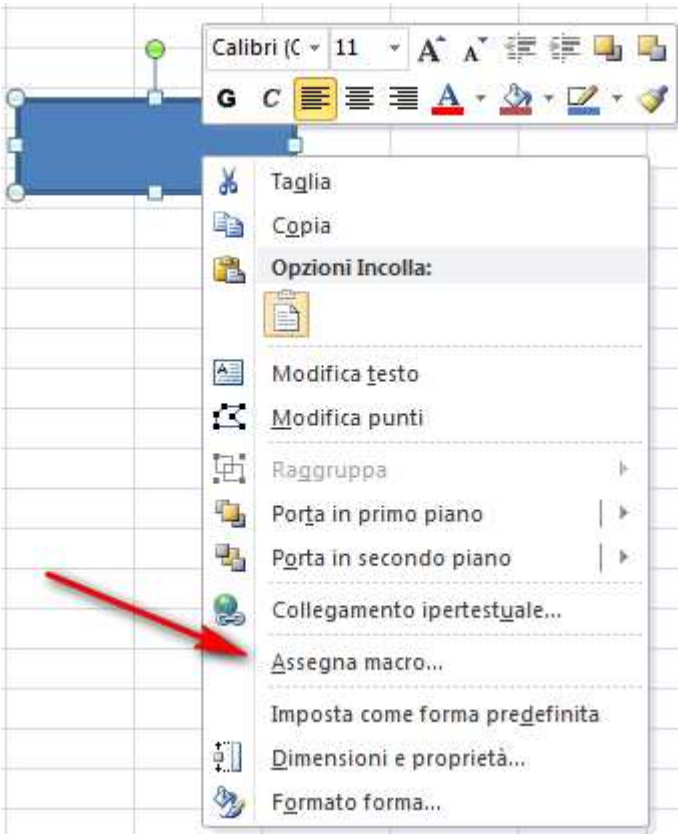

Fig. 10

Fine Nota

Possiamo ora mandare in esecuzione la macro stessa semplicemente cliccando sul pulsante appena creato, adesso il foglio è sparito, come facciamo a farlo riapparire? Basta creare una nuova macro che assoceremo ad un altro pulsante, con le stesse modalità di costruzione della precedente:

```
Sub Mostra_Foglio()
Worksheets(2).Visible = True
MsgBox "Il Foglio 2 è Ricomparso"
End Sub
```

I commenti sono superflui, il codice è esattamente identico al precedente solo che la proprietà **visible** è stata ora posta a **True** (Vero) ed in tal caso il Foglio 2 riappare.