- Alessandra Salvaggio -

Microsoft Excel 2013

macro e VBA

Nozioni e suggerimenti per la programmazione di macro e VBA >>

Creazione di componenti aggiuntivi >>

Sintassi, ottimizzazione, controllo del codice >>

Progetti pratici illustrati passo passo >>

EDIZIONI FAG MILANO

*pro Digital Life Style

*pro
Digital*LifeStyle

Microsoft
Excel 2013
macro e VBA

Alessandra Salvaggio

EDIZIONI
FAG
MILANO

Microsoft Excel 2013 macro e VBA

Autrice: Alessandra Salvaggio

Collana:

*pro
DigitalLifeStyle

Publisher: Fabrizio Comolli
Editor: Marco Aleotti
Progetto grafico: Roberta Venturieri
Impaginazione: Eleonora Moroni
Coordinamento editoriale, prestampa e stampa: escom - Milano
Immagine di copertina: © 4designersart | Dreamstime.com

ISBN: 978-88-6604-366-9

Via G. Garibaldi 5 – 20090 Assago (MI) - www.fag.it

Finito di stampare in Italia presso Press Grafica - Gravellona Toce - VB nel mese di ottobre 2013

Sommario

INTRODUZIONE A VBA 11

Parte 1

1. PRIMI PASSI: GESTIRE IL CONTENUTO DELLE CELLE 27
2. LE ISTRUZIONI CONDIZIONALI E L'INTERAZIONE CON L'UTENTE 33
   Scrivere il codice con ordine 35
3. UN CICLO FOR ... NEXT 37
   Selezionare una cella 38
   Comporre un messaggio: unire un testo e una variabile 38
   Interrompere un ciclo 40
4. UN CICLO FOR EACH ... NEXT 41
5. UTILIZZARE UN FORM E I RELATIVI CONTROLLI 43
   Gestiamo un possibile errore 50

Parte 2

6. IL CONTROLLO CALENDARIO 55
   Excel 2007 59
   Aggiungere il controllo date picker a Excel 2007 61
7. CALCOLARE I GIORNI LAVORATIVI 63
8. LIMITARE LE AZIONI DELL'UTENTE 69
9. IMMETTERE UN PARAMETRO 73
10. VERIFICARE ALCUNE DATE 77

Parte 3

11. INSERIRE UN DATO AUTOMATICAMENTE 83

12. CONTARE IL NUMERO DI CELLE PIENE IN UNA COLONNA 87

13. NASCONDERE LE SEGNALAZIONI DI ERRORE 91
Nascondere gli errori 94

14. GESTIRE GLI ERRORI 99

Parte 4

15. SPOSTARE LE RIGHE 105
La posizione dei pulsanti 109

16. CERCARE E VISUALIZZARE DEI DATI DI UN ELENCO IN UN FORM 113

17. CANCELLARE I DATI DI UN ELENCO 119

18. INTERAGIRE CON L'UTENTE TRAMITE LA FUNZIONE MSGBOX 121

19. UNA ROUTINE SUB ESTERNA 125
Passare un valore alla routine 127

20. AGGIORNARE I DATI 129

21. AGGIUNGERE I DATI 139

22. LE CASELLE COMBINATE 143

23. VISUALIZZARE UN'IMMAGINE NEL FORM 147

24. FILTRARE E STAMPARE I DATI 151

25. USARE IL FILTRO AVANZATO 157

26. COLLEGAMENTI IPERTESTUALI 163

Parte 5

27. SELEZIONARE DATI NON CONTIGUI 173

28. GENERARE UN GRAFICO 177

29. SCEGLIERE I DATI DI ORIGINE DEL GRAFICO 183
Inserire il comando per generare il grafico sulla barra di accesso 187

30. AGGIUNGERE UN NUOVO FOGLIO 191
Creare funzioni personalizzate 193
Quando un anno è bisestile 196
Aggiungere un nuovo foglio già in parte compilato 197

Parte 6

31. IMPORTARE UN FILE DI TESTO 209

32. I DELIMITATORI DEI DECIMALI E DELLE MIGLIAIA 219

33. GESTIRE LE DATE 223

34. IMPORTARE DAL WEB 227

Parte 7

35. RILEVARE LA MODIFICA DI UNA CELLA E APRIRE UN FILE ESISTENTE 235

36. APRIRE UN FILE PROTETTO DA PASSWORD E SALVARE CODICE VBA IN UN MODELLO 243
Salvare il file come modello 248

37. SALVARE UN FILE E CONVERTIRLO IN PDF 253
Salvare un file 256
Convertire un file in formato PDF 259

38. INVIARE VIA E-MAIL 261
Verificare un indirizzo e-mail 266

Parte 8

39. CONSOLIDARE 269

40. CONSOLIDARE INTERVALLI IN FILE DIVERSI 275

Parte 9

41. CREARE UN INDICE DEI FOGLI 283
Aggiornare l'indice 286

42. ORDINARE I FOGLI DI LAVORO IN BASE AL LORO NOME 291
L'ordine alfabetico 292
Spostare le righe 294

43. PERSONALIZZARE LA BARRA MULTIFUNZIONE 295

44. CREARE UN COMPONENTE AGGIUNTIVO 307
Attivare il componente aggiuntivo 309

45. UN MODULO DI CLASSE PER GESTIRE GLI EVENTI DI UN COMPONENTE AGGIUNTIVO 313
Gestire i pulsanti che spostano le righe 321

Parte 10

46. ESTRARRE I DATI UNIVOCI DA UN INTERVALLO DI PIÙ COLONNE 327
Ridimensionare l'array in funzione del numero di celle presenti nell'intervallo da cui estrarre i dati univoci 334

47. VERIFICARE SE DUE INTERVALLI SI INTERSECANO 337

48. RILEVARE L'INPUT DELL'UTENTE CON INPUT BOX 341
Cosa succede se l'utente non seleziona un intervallo? 344

49. COMPONENTE AGGIUNTIVO 347

Parte 11

50. VERIFICA DELLA PARTITA IVA 353

51. VERIFICA DEL CODICE FISCALE 357

52. VERIFICA DEL CODICE IBAN 365
Il componente aggiuntivo e la sua barra personalizzata 369

Parte 12

53. CREARE UNA LIBRERIA DI FUNZIONI 373
Inserire la funzione personalizzata da interfaccia, mostrare indicazioni sui parametri e sulla funzione stessa 378

Parte 13

54. IL DEBUG DEL CODICE 385
Gli strumenti di debugging 388

55. MIGLIORARE L'EDITOR 391
I blocchi di commento 391
Smart Indenter 393
VBETools 394
MZ-Tools 396

56. PROTEGGERE IL PROPRIO CODICE 399
Cancellare il codice 400

57. REGISTRARE LE MACRO: PER ANDARE AVANTI DA SOLI 407

INDICE ANALITICO 411

Introduzione a VBA

La suite da ufficio Microsoft Office è uno dei software più diffusi al mondo. Molti ne apprezzano le potenzialità, ma tanti ignorano che è possibile estenderne le capacità per adeguarle alle proprie esigenze. Tutto questo è possibile grazie a VBA, o Visual Basic for Applications, un linguaggio di programmazione appartenente alla famiglia del Visual Basic.

Chi conosce Visual Basic 6 si troverà a proprio agio con Visual Basic for Applications, in tutto affine al fratello maggiore. L'unica differenza è data dalle istruzioni specifiche per operare sui prodotti Office.
Anche chi è alla prima esperienza di programmazione potrà trovare in questo libro spunti interessanti per incrementare le capacità e la flessibilità dei propri file Excel.
Si chiede solamente una discreta conoscenza di Excel. Per il resto le istruzioni saranno date passo passo, in modo da guidare tutti a realizzare i progetti proposti dal libro.
Sul sito Internet: www.sos-office.it/libro-excel2013vba.html troverete tutti i file necessari per seguire gli esercizi. Si tratta di vari file che corrispondono ai progetti realizzati nel libro; a ogni progetto è dedicata una sezione del volume.
Questo testo si basa sulla versione 2013 di Office, ma la maggior parte degli esempi proposti dovrebbe funzionare correttamente anche con le versioni precedenti.

Gestire i file che contengono codice VBA

Office 2013 (come già Office 2010 e 2007), dunque anche Excel 2013, distingue i file che contengono codice VBA dagli altri file con un'estensione e un'icona diverse (Figura 0.1).

Figura 0.1 - A sinistra l'icona di un file che non contiene codice VBA e a destra quella di uno che lo contiene.

I normali file di Excel hanno estensione .xlsx, mentre quelli che contengono codice VBA hanno estensione .xlsm. Anche il formato di salvataggio è diverso.
Quando salvate un file che contiene codice VBA (le porzioni di codice vengono definite Macro) occorre indicare a Excel che deve salvarlo usando il formato **Cartella di lavoro con attivazione di macro di Excel** (Figura 0.2).

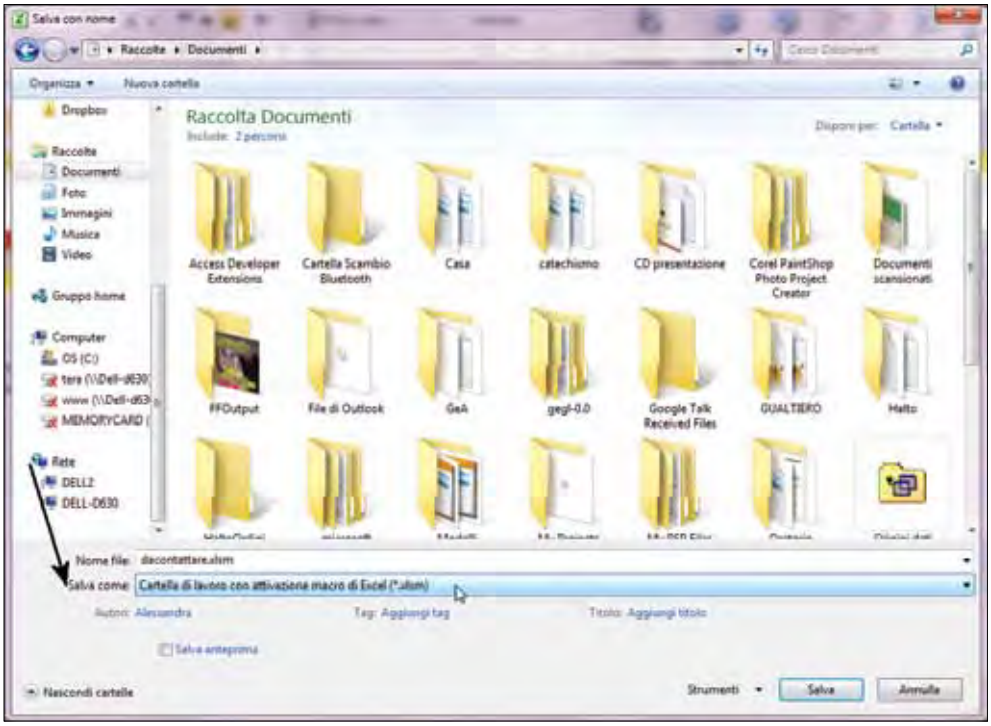

Figura 0.2 - Salvare un file in formato Cartella di lavoro con attivazione di macro di Excel.

Qualora tentaste di salvare il file nel formato tradizionale, Excel vi avviserebbe che non è possibile salvare le caratteristiche del vostro file nel formato che avete scelto (Figura 0.3).

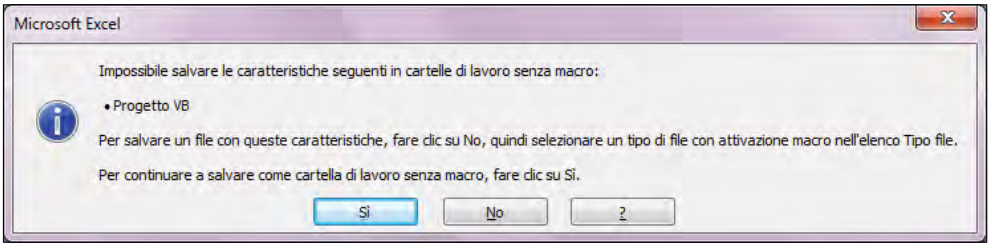

Figura 0.3 - Excel avvisa che occorre un formato di file adatto alle macro VBA.

Nella finestra di avviso scegliete **No**. In questo modo Excel vi mostrerà la finestra **Salva con nome**, nella quale potrete scegliere il formato corretto per conservare il vostro codice VBA.
Se, invece, scegliete **Sì** e salvate comunque in formato .xlsx, il codice che avete inserito nel file non sarà in alcun modo eseguibile né utilizzabile.
La differenziazione del formato dei file che contengono codice VBA si è resa necessaria anche per il fatto che il codice VBA può essere pericoloso.
L'esecuzione di codice che, di fatto, compie delle operazioni all'insaputa dell'utente, può comportare dei rischi: non di rado, si sono trovati documenti Office che, all'interno del VBA, nascondevano dei virus.
Per questo, oltre a differenziare il formato, Excel vi permette di stabilire come gestire i singoli file che contengono codice.
Volendo, è possibile fare in modo che Excel blocchi l'esecuzione di tutto il codice, ma se si ha bisogno del VBA si tratta di un'impostazione troppo drastica. D'altro canto, lasciare che Excel esegua qualsiasi codice, anche quello non scritto da voi, può essere pericoloso. La soluzione giusta consiste nel fare in modo che, all'apertura di tutti i file che contengono codice, Excel vi chieda se eseguirlo o meno.
Per determinare queste impostazioni occorre che, sulla barra multifunzione, sia visibile la scheda **Sviluppo**. Se non lo fosse, portatevi alla scheda **File** e, nell'area grigia a sinistra, fate clic su **Opzioni**. Visualizzerete la finestra **Opzioni**. Qui, portatevi alla scheda **Personalizzazione barra multifunzione** (Figura 0.4).

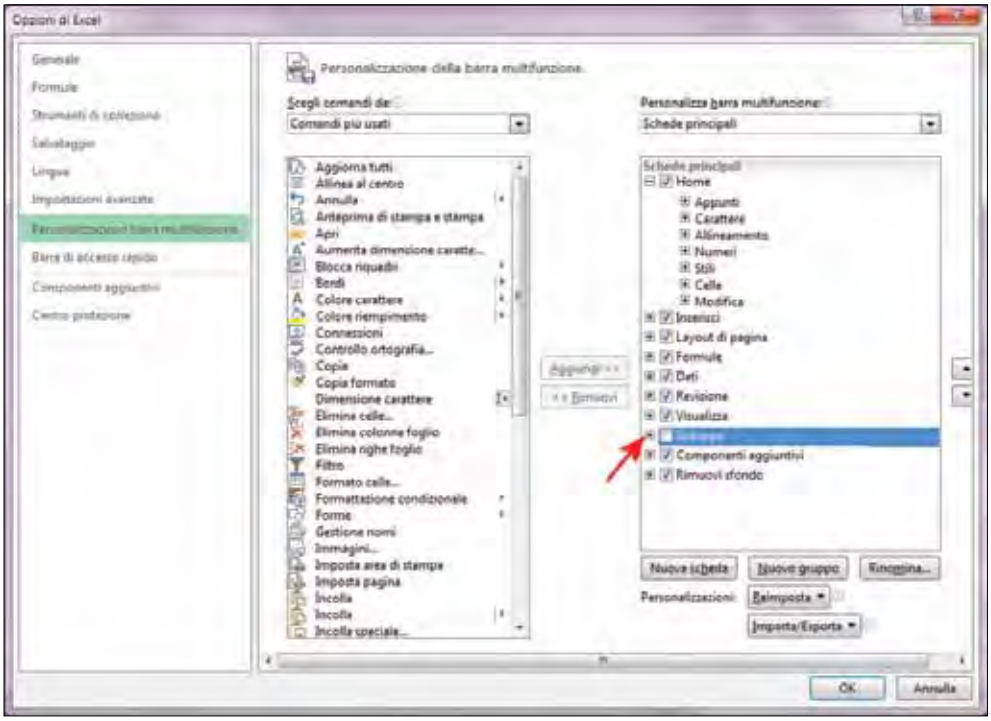

Figura 0.4 - Aggiungere la scheda Sviluppo alla barra multifunzione.

Nel riquadro a destra, selezionate la scheda **Sviluppo**. Poi premete **OK** per tornare al foglio di lavoro.

NOTA

Chi lavora con Excel 2007, per visualizzare la scheda **Sviluppo** deve aprire il menu del pulsante Microsoft Office e premere il pulsante **Opzioni di Excel**. Apparirà la finestra **Opzioni di Excel**. Nella sezione **Impostazioni generali** occorre mettere un segno di spunta alla voce **Mostra scheda Sviluppo sulla barra multifunzione**.
In Excel 2003, invece, le opzioni relative alla sicurezza delle macro si trovano nel menu **Strumenti > Macro**.

Portatevi alla scheda **Sviluppo**. Qui, nel gruppo **Codice**, premete il pulsante **Sicurezza macro** (**Protezione macro**, con Excel 2007). Si aprirà la finestra **Centro protezione** (Figura 0.5).

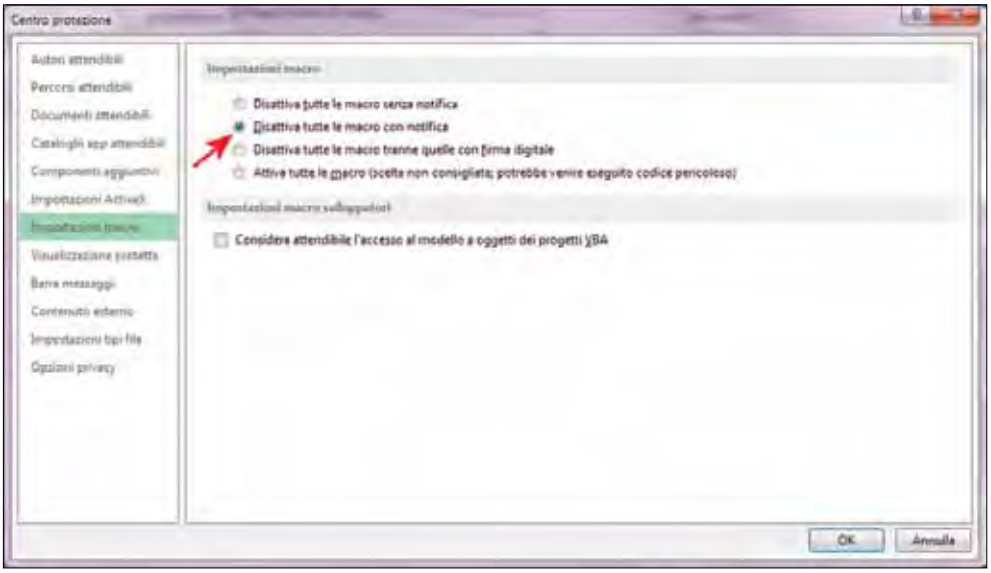

Figura 0.5 - La finestra Centro protezione.

Le opzioni disponibili sono:

- **Disattiva tutte le macro senza notifica**. Utilizzate questa impostazione se le macro non sono considerate attendibili. Tutte le macro nei documenti e gli avvisi di protezione per le macro verranno disattivati;
- **Disattiva tutte le macro con notifica**. Utilizzate questa opzione se volete disattivare le macro, ma desiderate anche ricevere messaggi di avviso nel caso siano presenti delle macro. In questo modo è possibile scegliere se attivare o meno le singole macro;
- **Disattiva tutte le macro tranne quelle con firma digitale**. Questa impostazione è simile all'opzione precedente, con la differenza che la macro può essere eseguita se è firmata da un editore attendibile che è già stato considerato tale. In caso contrario, si riceverà una notifica e sarà possibile scegliere se attivare le singole macro firmate o considerare attendibile l'editore. Tutte le macro senza firma vengono disattivate senza notifica;
- **Attiva tutte le macro**. Tutte le macro vengono eseguite senza che ne siate avvisati. È un'impostazione pericolosa e vi sconsigliamo di adottarla.

Noi vi consigliamo l'opzione predefinita **Disattiva tutte le macro con notifica**. In questo modo, quando aprite un file contenente codice VBA, Excel vi avvisa che il file include contenuto potenzialmente pericoloso e che questo è stato bloccato (Figura 0.6).

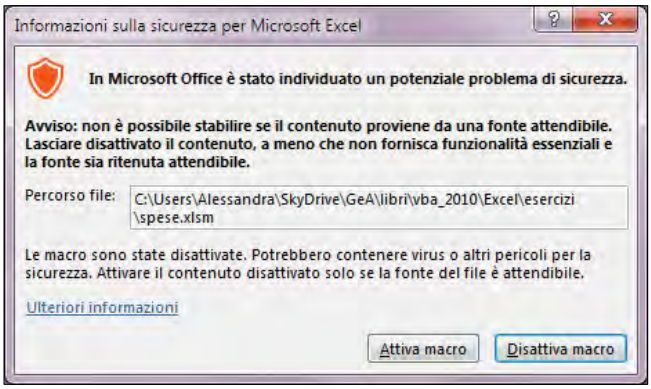

Figura 0.6 - Excel notifica la presenza di codice potenzialmente pericoloso.

Se volete attivare il codice VBA per poterlo utilizzare, premete il pulsante **Abilita contenuto**.

In alternativa a questa procedura, potete decidere di salvare tutti i file con il codice VBA in una cartella e definire quella cartella come **percorso attendibile**.

Nella finestra **Centro protezione** (Figura 0.5), nell'area a destra, selezionate la categoria **Percorsi attendibili** (Figura 0.7).

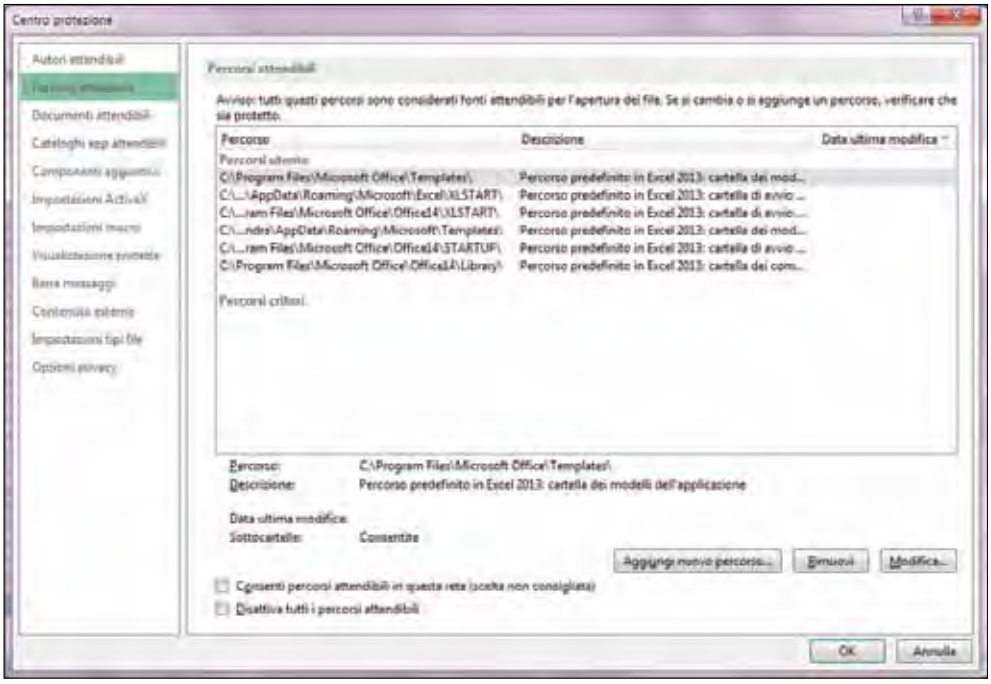

Figura 0.7 - La finestra Protezione alla scheda Percorsi attendibili.

Premete il pulsante **Aggiungi nuovo percorso** e individuate la cartella che contiene i file con il codice VBA.
Il percorso apparirà nella parte alta della scheda **Percorsi attendibili** (Figura 0.7). Chiudete la finestra. Ora potrete tranquillamente aprire i file di questa cartella e il codice VBA sarà automaticamente attivato.
Oltre ai percorsi attendibili, che Excel 2010/13 hanno ereditato da Excel 2007, le versioni di Excel più recenti dispongono anche dei **Documenti attendibili**.
Vediamo di cosa si tratta. Quando apriamo un documento che contiene macro o oggetti ActiveX, come abbiamo visto, un messaggio ci informa che questo file contiene elementi potenzialmente pericolosi (Figura 0.6).
Abbiamo detto che per attivare il contenuto potenzialmente pericoloso basta scegliere di abilitarlo con l'apposito pulsante.
Questa procedura valeva anche per le versioni di Excel precedenti alla 2010/13.
L'autorizzazione all'uso di questi file, però, andava ridata ogni volta che si apriva il file.
A partire da Excel 2010, una volta che il file è stato autorizzato diventa un documento attendibile e, quando viene riaperto, le volte successive, non occorre più autorizzare il contenuto pericoloso esplicitamente.
Naturalmente, c'è un modo per revocare questa "fiducia" concessa ai file. Purtroppo, però, non lo si può fare un file alla volta, ma è solo possibile cancellare l'elenco dei file ritenuti attendibili tutti in una volta sola. Per farlo, basta portarsi alla finestra della Figura 0.5 e attivare la scheda **Documenti attendibili** (Figura 0.8). Qui occorre premere il pulsante **Cancella**. L'elenco dei file ritenuti attendibili viene rimosso. Bisognerà autorizzare nuovamente il contenuto di ogni singolo file.

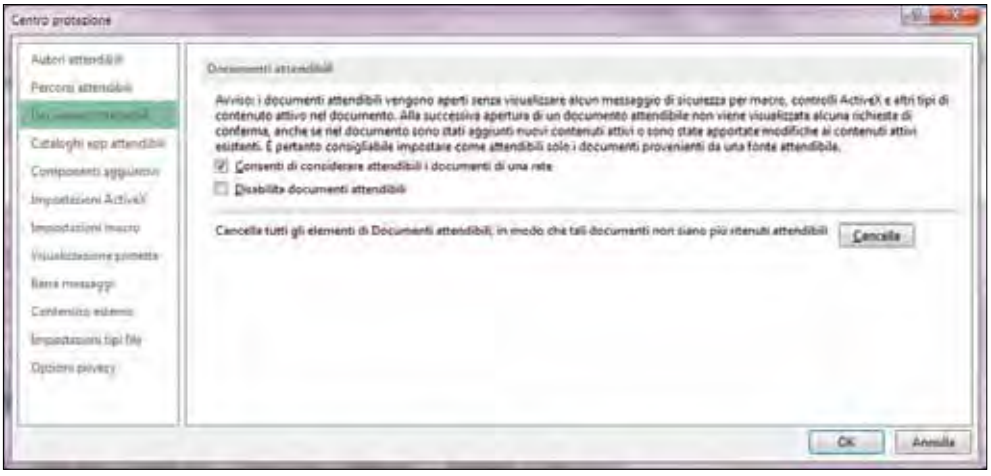

Figura 0.8 - La scheda Documenti attendibili.

È disponibile anche un'opzione meno drastica, ossia disabilitare i documenti attendibili temporaneamente. Basta selezionare la casella **Disabilita documenti attendibili**, sempre nella scheda **Documenti attendibili** (Figura 0.8). Per ogni documento potenzialmente pericoloso verrà richiesta l'autorizzazione, ma l'elenco dei documenti attendibili sarà comunque mantenuto e verrà nuovamente utilizzato appena viene deselezionata l'opzione **Disabilita documenti attendibili**.

Ambiente di lavoro

Dopo questa lunga, ma doverosa premessa sulla gestione dei file che contengono il codice VBA, passiamo a descrivere l'ambiente in cui vi troverete a lavorare.
Naturalmente avrete a che fare con il foglio di lavoro Excel, con il quale dovreste già avere familiarità, ma ancora di più con l'editor di Visual Basic (pressoché identico nelle diverse versioni di Excel), a cui dedicheremo le prossime pagine. Ricordate che tutto il codice VBA che scriverete vive unicamente all'interno del programma Office, nel nostro caso Excel, che state utilizzando. L'unico file con cui avrete a che fare sarà il file Excel, che contiene sia i fogli di lavoro a cui siete normalmente abituati sia il codice VBA.
Per aprire l'editor di Visual Basic, all'interno di Excel (del resto non può vivere all'esterno delle applicazioni Office), portatevi alla scheda **Sviluppo** (poco fa avete imparato a visualizzarla) e, nel gruppo **Codice**, premete il pulsante **Visual Basic**. Si aprirà l'editor di Visual Basic.

> **NOTA** Qualsiasi versione di Excel usiate, per visualizzare l'editor di Visual Basic potete anche premere contemporaneamente **Alt** e **F11**.

L'editor di Visual Basic (Figura 0.9) si compone di tre sezioni principali:

1. **Area del codice**. La prima volta che avviate l'editor, quest'area si presenta vuota. In seguito, qui vedrete e scriverete il codice VBA;
2. **Area del progetto**. In questa sezione vedete tutti gli elementi che costituiscono un progetto. Alla prima apertura, vedete i tre fogli Excel e la cartella di lavoro (ThisWorkbook). A ciascuno degli oggetti presenti nel progetto può essere associato del codice (nel parleremo nel paragrafo successivo);
3. **Area delle proprietà**. Qui trovate le proprietà di tutti gli elementi che avete inserito nel vostro progetto. Discuteremo delle proprietà dei singoli elementi più avanti, a mano a mano che li utilizzeremo. Qui accenneremo solo alla proprietà più importante, ossia **Name**. Attraverso questa proprietà ogni oggetto viene identificato in maniera univoca. È molto importante che assegniate dei nomi

significativi a tutti gli oggetti del vostro progetto: solo così potrete richiamarli in modo semplice e, soprattutto nei progetti complessi, potrete ricordarvi la funzione di ciascun elemento. Più avanti discuteremo delle convenzioni da adottare per assegnare i nomi agli oggetti.

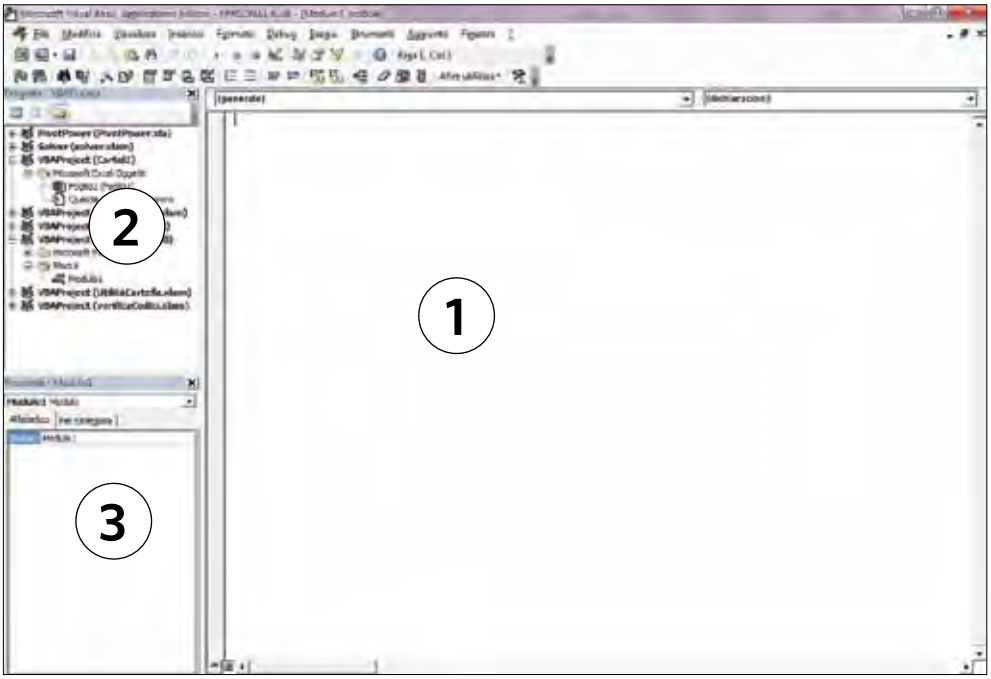

Figura 0.9 - L'editor di Visual Basic.

Gli elementi di un progetto Visual Basic

Un progetto VBA si compone di numerosi elementi a ciascuno dei quali può essere associato del codice Visual Basic. La Figura 0.10 mostra l'**Area del progetto** relativa a un progetto composto da vari elementi. Li analizzeremo uno per uno.

Innanzitutto, troviamo la sezione **Microsoft Excel Oggetti**, in cui sono elencati tutti i fogli contenuti nella cartella di lavoro e la cartella di lavoro stessa. Accanto al numero di ciascun foglio, tra parentesi, è indicato il suo nome, che è lo stesso che si vede sulle linguette in basso, all'interno di Excel.

A ogni foglio, o alla cartella di lavoro, può essere associato del codice. Facendo doppio clic sull'oggetto (foglio o cartella di lavoro che sia), nell'area del codice viene visualizzato un foglio vuoto pronto per essere riempito (Figura 0.11); impareremo a farlo dal prossimo capitolo.

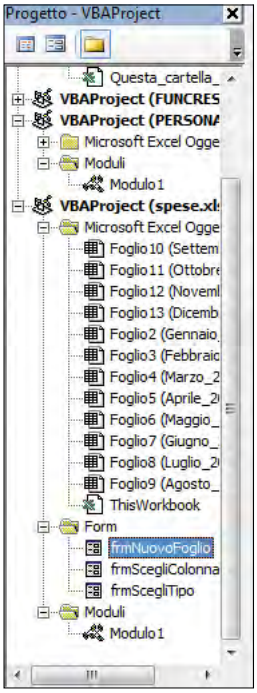

Figura 0.10 - L'Area del progetto.

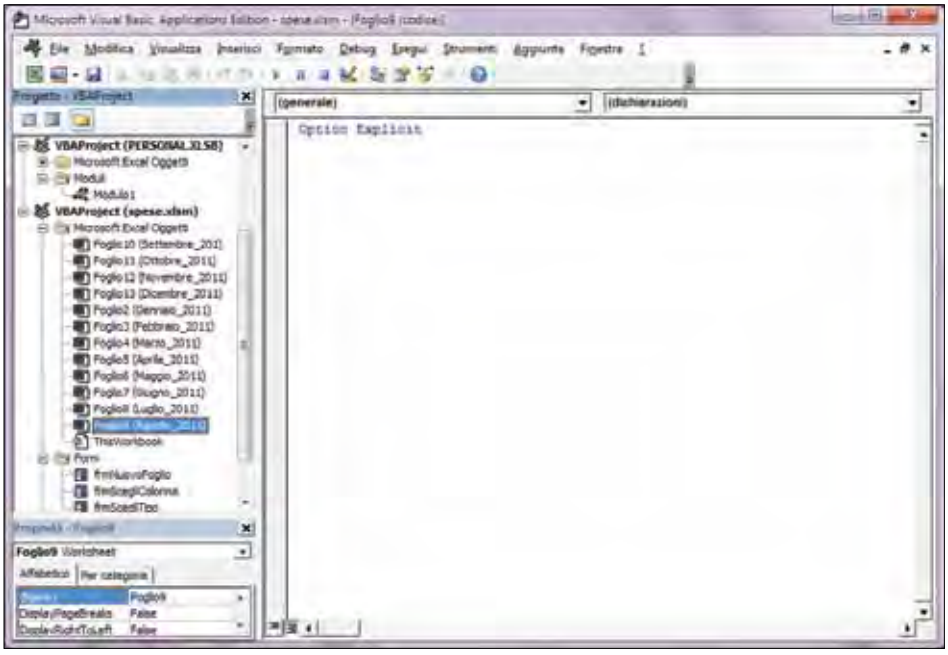

Figura 0.11 - Il codice associato al Foglio 9.

Oltre agli oggetti di Excel, all'interno di un progetto VBA potrete trovare (e quindi utilizzare) i **form** (Figura 0.12), ossia finestre di dialogo che i vostri utenti potranno usare per effettuare scelte, selezionare ciò che interessa ecc.

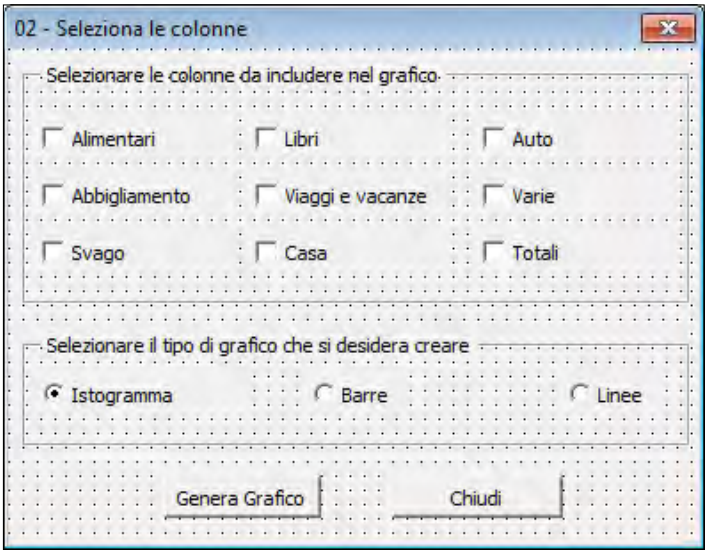

Figura 0.12 - Un form aperto all'interno dell'editor di VBA.

A ogni form è associato un foglio di codice, che normalmente ospita le istruzioni VBA necessarie al funzionamento del form stesso. I form, all'interno della finestra del progetto, sono raggruppati nella cartella Form.
Sia sul form, sia direttamente all'interno di un foglio di lavoro è possibile inserire una serie di controlli, in tutto simili a quelli che si trovano nelle finestre di dialogo di Windows e dei più comuni programmi. Tra i più usati trovate:

- **Caselle di testo** o **TextBox**, dove l'utente può digitare del testo;
- **Etichette** o **Label**: spazi in cui potete scrivere del testo; per esempio, per spiegare all'utente cosa deve scrivere in una casella di testo;
- **Caselle di controllo** o **CheckBox** : caselle rettangolari, di solito presentate in gruppi, in cui l'utente può selezionare **più di una** opzione;
- **Pulsanti di opzione** o **Option Button** : pulsanti circolari, di solito presentati in gruppi, in cui l'utente può selezionare **una sola** opzione;
- **Caselle combinate** o **Combo Box**: caselle di testo unite a elenchi a discesa in cui l'utente può selezionare l'opzione che desidera o digitare del testo. Nelle applicazioni Office, un esempio di casella combinata è quella utilizzata per la scelta del tipo di font con cui si vuole scrivere;

- **Liste** o **ListBox**: caselle di riepilogo con un elenco di dati fra cui l'utente può scegliere;
- **Pulsanti di comando**: normali pulsanti premendo i quali si avvia un'azione.

Questi e altri controlli si inseriscono nei fogli Excel e nei form attraverso la **Casella degli strumenti** (Figura 0.13), che potete attivare, all'interno di Excel, premendo il pulsante **Inserisci** nel gruppo **Controlli** nella scheda **Sviluppo** della barra multifunzione. Nell'editor di VBA potete aprire la casella degli strumenti scegliendo **Visualizza à Casella degli strumenti**.

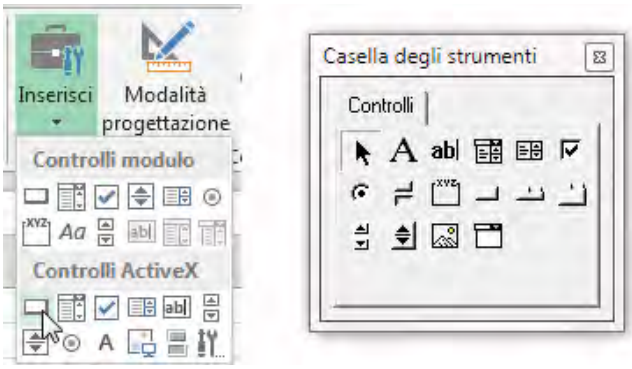

Figura 0.13 - A sinistra la casella degli strumenti in Excel, a destra nell'editor di VBA.

Oltre ai fogli di lavoro Excel e ai form con tutti i controlli che possono ospitare, un progetto VBA può contenere anche dei **Moduli**, raggruppati nella cartella Moduli.
Un modulo non può includere controlli come i fogli di lavoro e i form, ma contiene esclusivamente codice. In genere si tratta di porzioni di codice generali che possono essere richiamate e utilizzate in più punti del progetto (se il concetto non vi è chiaro non preoccupatevi troppo, prima della fine del libro capirete...).

Convenzioni per i nomi

Prima di cominciare a scrivere il nostro primo progetto VBA occorre precisare che, anche se ci troviamo all'interno di un'applicazione Office, si tratta sempre di programmazione e l'ordine e la precisione sono molto importanti se non si vogliono fare pasticci.
La prima regola fondamentale è assegnare a ciascun elemento del vostro progetto un nome "parlante", ossia un nome che, anche a distanza di tempo, vi permetta di capire con che tipo di oggetto avete a che fare e a che cosa serve. Credete, non è una cosa da poco quando si utilizzano molti oggetti e controlli.

Normalmente si adotta un sistema di denominazione composto da un prefisso che indichi il tipo di oggetto e un nome vero e proprio, che descriva l'oggetto stesso. Fra il prefisso e il nome non può esserci lo spazio vuoto (non è ammesso dal VBA, così come non sono ammessi i segni di punteggiatura): per questo, per identificare velocemente le due parti, si scrive la prima lettera del nome vero e proprio con la lettera maiuscola. Facciamo qualche semplice esempio: una casella di testo in cui un utente deve inserire il suo cognome si dovrebbe chiamare **txtCognome**; un form in cui un utente deve inserire i suoi dati anagrafici non potrà che chiamarsi **frmAnagrafica**. Semplice no? È bene abituarsi fin dall'inizio a nominare correttamente gli oggetti, tutto poi vi verrà più facile. Qui di seguito troverete un elenco dei principali prefissi in uso. Tenete presente che i prefissi sono tratti dai nomi inglesi dei diversi controlli e oggetti.
Per i form e i moduli i prefissi sono rispettivamente **frm** e **mod**. Per i diversi tipi di controlli, invece, abbiamo:

- **txt** per le caselle di testo;
- **lbl** per le etichette;
- **cbo** per le caselle combinate;
- **lst** per le liste;
- **cmd** per i pulsanti di comando;
- **opt** per i pulsanti di opzione;
- **chk** per le caselle di controllo;
- **img** per le immagini.

Modello a oggetti

Occorre fare una rapida premessa sui principi su cui si fonda il VBA, per poter fissare quella terminologia che poi utilizzeremo nel corso del libro. VBA è un **linguaggio a eventi** che segue (almeno in parte) i principi della programmazione **orientata agli oggetti**.
Cerchiamo di chiarire, uno per volta, questi due concetti. Un linguaggio di programmazione a eventi è un linguaggio in cui le azioni vengono compiute quando si verifica un evento. In parole povere, il codice viene eseguito quando l'utente fa qualcosa, come premere un pulsante, scrivere in una casella di testo, uscire da una casella di testo... Tutte queste azioni sono eventi. A ogni evento può essere associato un codice, come dire: all'evento "bussare" viene attivata l'azione "aprire".
Ogni evento è associato a un oggetto (e qui passiamo al secondo concetto): questo significa che non diremo "io busso alla porta", ma "porta_bussano". Qui abbiamo anticipato un po' la sintassi di VBA, infatti il nome di un oggetto e l'evento sono separati da un trattino basso o, in inglese, **underscore** (_).

Ma quali sono gli oggetti? Gli oggetti sono i fogli di lavoro, i form, i controlli come le caselle di testo, i pulsanti ecc.
Gli oggetti, oltre a essere associati a eventi, possono avere delle proprietà, ossia caratteristiche che possiamo leggere e modificare. Per esempio, l'oggetto form ha come proprietà il titolo (caption). In VBA dirò che **frmAnagrafica.caption** = **"Dati anagrafici"**, il che vuol dire che la proprietà titolo dell'oggetto form **frmAnagrafica** è "Dati anagrafici", che è un po' come se dicessi **capelli.colore** = "biondi", ossia i miei capelli sono biondi. Ogni proprietà è unita all'oggetto a cui si riferisce da un punto.
Oltre alle proprietà, gli oggetti hanno dei metodi, ossia delle azioni che vengono compiute sugli oggetti stessi. Per esempio, l'oggetto form ha il metodo Hide, che nasconde il form stesso: dirò **frmAnagrafica.hide** per indicare a VBA di nascondere il form **frmAnagrafica** (in seguito al verificarsi di un evento che avremo specificato), che è un po' come dire **capelli.taglia**. Come vedete, anche i metodi sono uniti agli oggetti da un punto.
Questa situazione può essere complicata dal fatto che gli oggetti possono essere contenuti gli uni dentro gli altri: **frmAnagrafica.txtCognome.text** = **"Salvaggio"** significa che la proprietà text (ossia contenuto testuale) della casella di testo **txtCognome**, all'interno del form **frmAnagrafica**, è uguale a Salvaggio, un po' come dire **Alessandra.capelli** = **"biondi"**, cioè i capelli della persona Alessandra sono biondi.
La gerarchia degli oggetti viene descritta da sinistra a destra, ovvero l'oggetto citato per primo, quello più a sinistra (**frmAnagrafica** nell'esempio precedente) è quello che contiene gli altri oggetti (**txt.Cognome**), quelli più a destra.
Non sempre bisogna elencare tutti gli oggetti uno dentro l'altro: per esempio, se ci si trova all'interno del modulo associato al form **frmAnagrafica**, non ci sarà bisogno di scrivere **frmAnagrafica.txtCognome.text** = **"Salvaggio"**, ma basterà **txtCognome.text** = **"Salvaggio"**.

Autore e utente

Nel libro parleremo dell'utente dei file realizzati con VBA, distinguendolo da chi ha creato il file. Nella realtà, molto spesso, le due figure coincideranno e voi scriverete i vostri progetti VBA per il vostro uso personale. La distinzione fra utente e autore dei progetti è comunque una distinzione di ruoli.
Quando create i vostri progetti, siete autori, quando li usate diventate utenti.
Quando siete autori dovete pensare a creare un progetto semplice da usare e che non presenti problemi per l'utente, anche se l'unico vostro utente siete voi stessi. Non è certo simpatico ritrovarsi con un progetto che funziona male e che, durante l'uso, deve essere sistemato.