Capitolo 42: Lavorare con gli eventi di Excel

1. Comprendere gli Eventi in Excel VBA

a. Cosa sono gli eventi

In Excel VBA, un evento è un'azione o un'accadimento che Excel può 'intercettare' e al quale può reagire con l'esecuzione di una macro. Gli eventi sono associati a oggetti come fogli di lavoro, cartelle di lavoro, controlli ActiveX e UserForm.

b. Tipi comuni di eventi

- Workbook_Open: quando si apre una cartella di lavoro
- Worksheet_Change: quando cambia il contenuto di una cella
- Worksheet_SelectionChange: quando si seleziona un'altra cella
- CommandButton_Click: quando si clicca un pulsante
- UserForm_Initialize: quando una UserForm viene aperta

c. Dove scrivere gli eventi

Gli eventi di oggetti come Workbook o Worksheet si scrivono nei rispettivi moduli nel VBA Project.
Per scrivere eventi di controlli ActiveX o UserForm, si lavora direttamente nel modulo del controllo o della maschera.

d. Sintassi degli eventi

Esempio - Evento Change di un foglio:

```
Private Sub Worksheet_Change(ByVal Target As Range)
    MsgBox "Hai modificato " & Target.Address
End Sub
```

Esempi pratici

Mostrare un messaggio quando si apre la cartella

```
Private Sub Workbook_Open()
    MsgBox "Benvenuto nel file!"
End Sub
```

Avvisare se viene modificata la cella A1

```
If Target.Address = "$A$1" Then MsgBox "Hai cambiato A1"
```

Catturare un clic su un CommandButton

```
Private Sub CommandButton1_Click()
    MsgBox "Hai cliccato il pulsante."
End Sub
```

Impostare il focus in una UserForm

```
Private Sub UserForm_Initialize()
    TextBox1.SetFocus
End Sub
```

Reagire alla selezione della cella B2

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    If Target.Address = "$B$2" Then MsgBox "Hai selezionato B2"
End Sub
```

Esercizi

Scrivi un evento che mostri un messaggio se viene modificata la cella C3
Usa Worksheet_Change e controlla Target.Address.
Crea un pulsante con un evento Click che scriva un valore in A1
Scrivi CommandButton1_Click e usa Range().Value
Implementa un evento Workbook_BeforeClose che chieda conferma all'utente
Usa If MsgBox(...) = vbNo Then Cancel = True
Crea un evento che modifichi il colore di sfondo di una cella selezionata
Usa Worksheet_SelectionChange e Target.Interior.Color
Imposta il contenuto di un TextBox all'apertura della UserForm
UserForm_Initialize con TextBox1.Text = "Valore predefinito"

2. Inserire codice VBA per la gestione degli eventi Rel.1

a. Cos'è un gestore di eventi (Event Handler)

Un gestore di eventi è una subroutine speciale che viene eseguita automaticamente quando si verifica un evento specifico su un oggetto come un foglio di lavoro, una cartella di lavoro, un controllo ActiveX o una UserForm.

b. Dove inserire il codice

- Eventi di foglio di lavoro: fai doppio clic sul nome del foglio nel progetto VBA e seleziona l'evento dall'elenco a discesa.
- Eventi di cartella di lavoro: usa 'ThisWorkbook'.
- Eventi di controlli ActiveX: doppio clic sul controllo in modalità struttura.
- Eventi di UserForm: doppio clic su un controllo o sulla maschera stessa.

c. Struttura di un gestore di eventi

Un evento è sempre una Subroutine e ha parametri specifici. Esempio:

```
Private Sub Worksheet_Change(ByVal Target As Range)
    ' Codice da eseguire
End Sub
```

Questa subroutine si attiva quando il contenuto di una cella cambia.

Esempi pratici

Evento che si attiva quando cambia la cella A1

```
Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Address = "$A$1" Then MsgBox "Hai modificato A1"
End Sub
```

Evento Click di un pulsante

```
Private Sub CommandButton1_Click()
    MsgBox "Pulsante cliccato."
End Sub
```

Evento Initialize di una UserForm

```
Private Sub UserForm_Initialize()
    TextBox1.Text = "Inserisci il tuo nome"
End Sub
```

Evento di selezione cella B2

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    If Target.Address = "$B$2" Then MsgBox "Hai selezionato B2"
End Sub
```

Evento BeforeClose della cartella

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    If MsgBox("Vuoi davvero uscire?", vbYesNo) = vbNo Then Cancel = True
End Sub
```

Esercizi

Crea un evento che mostra un messaggio quando viene selezionata la cella C5
Usa Worksheet_SelectionChange e controlla Target.Address.
Scrivi il codice Click per un pulsante che inserisce la data corrente in A1
Usa Range("A1").Value = Date
Mostra un messaggio di benvenuto con l'evento Workbook_Open
Scrivi MsgBox "Benvenuto" in ThisWorkbook.
Imposta il valore di una TextBox all'avvio della UserForm
Usa TextBox1.Text = "Ciao" in UserForm_Initialize
Blocca la chiusura del file se una cella specifica è vuota
Usa Workbook_BeforeClose e controlla Range("A1").Value = ""

3. Inserire Codice VBA per la Gestione degli Eventi Rel.2

a. Cos'è un gestore di eventi (Event Handler)

Un gestore di eventi è una procedura che viene eseguita automaticamente in risposta a un evento generato da Excel o da un controllo ActiveX/UserForm. Gli eventi possono riguardare fogli, cartelle, controlli o maschere.

b. Come inserire codice VBA per eventi

- Apri l'editor VBA con Alt + F11
- Nel riquadro a sinistra, seleziona l'oggetto (es. Foglio1, ThisWorkbook, UserForm1)
- Dal menu a discesa in alto seleziona l'oggetto e l'evento
- Scrivi il codice nella procedura generata automaticamente

c. Struttura base di un gestore di evento

Esempio per un pulsante:

```
Private Sub CommandButton1_Click()
    MsgBox "Hai cliccato il pulsante."
End Sub
```

Esempi pratici

Evento Click su CommandButton

```
Private Sub CommandButton1_Click()
    MsgBox "OK cliccato."
End Sub
```

Evento Worksheet_Change per cella A1

```
If Target.Address = "$A$1" Then MsgBox "Modificato A1"
```

Evento UserForm_Initialize

```
Private Sub UserForm_Initialize()
    ComboBox1.AddItem "Opzione 1"
End Sub
```

Evento CheckBox_Change

```
If CheckBox1.Value Then MsgBox "Attivato"
```

Evento Workbook_Open

```
Private Sub Workbook_Open()
    MsgBox "File aperto."
End Sub
```

Esercizi

Scrivi un evento che aggiorna una cella al clic di un pulsante
Usa CommandButton1_Click per scrivere in A1.
Mostra un messaggio all'avvio del file
Inserisci codice in Workbook_Open.
Usa un evento per popolare una ComboBox all'avvio della UserForm
Aggiungi elementi in UserForm_Initialize.
Crea un evento che reagisce alla modifica della cella B2
Scrivi codice in Worksheet_Change.
Controlla se una CheckBox è selezionata e cambia il colore della cella
Usa CheckBox1_Change con Interior.Color.

4. Utilizzo degli eventi a livello di cartella di lavoro (Workbook-Level Events)

a. Utilizzo degli eventi a livello di cartella di lavoro

Gli eventi a livello di cartella di lavoro (Workbook) permettono di eseguire automaticamente codice VBA in risposta a determinate azioni dell'utente o del sistema, come l'apertura, la chiusura, il salvataggio del file o l'attivazione di fogli.

b. Utilizzo dell'evento Open

Esempio di codice:

```
Private Sub Workbook_Open()
    MsgBox "Benvenuto nel file Excel!"
End Sub
```

Questo evento si attiva automaticamente quando il file viene aperto.

c. Utilizzo dell'evento SheetActivate

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    MsgBox "Hai attivato il foglio: " & Sh.Name
End Sub
```

L'evento viene attivato ogni volta che un foglio diverso viene selezionato.

d. Utilizzo dell'evento NewSheet

```
Private Sub Workbook_NewSheet(ByVal Sh As Object)
    MsgBox "Hai creato un nuovo foglio: " & Sh.Name
End Sub
```

Viene attivato quando un nuovo foglio viene aggiunto alla cartella.

e. Utilizzo dell'evento BeforeSave

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
    If MsgBox("Vuoi davvero salvare?", vbYesNo) = vbNo Then Cancel = True
End Sub
```

Permette di bloccare o personalizzare il salvataggio della cartella.

f. Utilizzo dell'evento BeforeClose

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    If MsgBox("Sei sicuro di voler chiudere il file?", vbYesNo) = vbNo Then Cancel = True
End Sub
```

Questo evento consente di intercettare la chiusura del file e richiedere conferma.

Esempi pratici

Mostrare un messaggio di benvenuto all'apertura del file
Workbook_Open con MsgBox "Benvenuto!"
Avvisare l'utente su quale foglio è passato
Workbook_SheetActivate mostra Sh.Name
Confermare la creazione di un nuovo foglio
Workbook_NewSheet con MsgBox "Hai creato: " & Sh.Name
Bloccare il salvataggio se A1 è vuota
If Range("A1") = "" Then Cancel = True
Chiedere conferma prima della chiusura
Workbook_BeforeClose con MsgBox e Cancel = True se l'utente annulla

Esercizi

Scrivi un evento Open che imposti la data odierna in A1
Usa Range("A1") = Date
Mostra un messaggio se il foglio attivo è 'Dati'
Verifica Sh.Name = "Dati" in Workbook_SheetActivate

Blocca la chiusura se una cella contiene il valore 'Bloccato'
Controlla Range("B1") = "Bloccato" e imposta Cancel = True
Avvisa l'utente se salva un file con meno di 2 fogli
Usa If Worksheets.Count < 2 Then Cancel = True
Scrivi un evento che rinomina ogni nuovo foglio come 'Foglio_' & Data
Usa Workbook_NewSheet con Sh.Name = "Foglio_" & Format(Now, "yyyymmdd")

5. Lavorare con gli eventi dei fogli di lavoro (Worksheet Events)

a. Utilizzo degli eventi dei fogli di lavoro

Gli eventi dei fogli di lavoro consentono di eseguire codice in risposta a modifiche nelle celle, cambi di selezione, clic del tasto destro e altre interazioni dell'utente. Questi eventi sono molto utili per personalizzare il comportamento dei fogli.

b. Utilizzo dell'evento Change

L'evento Worksheet_Change si attiva ogni volta che una cella del foglio viene modificata.

Sintassi:

```
Private Sub Worksheet_Change(ByVal Target As Range)
    ' Codice
End Sub
```

c. Monitoraggio di un intervallo specifico

Per reagire solo a modifiche in un determinato intervallo:

```
If Not Intersect(Target, Range("A1:A10")) Is Nothing Then
    MsgBox "Hai modificato un valore nell'intervallo A1:A10"
End If
```

d. Utilizzo dell'evento SelectionChange

Questo evento si attiva quando l'utente seleziona una cella diversa.

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    ' Codice
End Sub
```

e. Utilizzo dell'evento BeforeRightClick

Si attiva quando l'utente clicca con il tasto destro del mouse.

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As Boolean)
    MsgBox "Hai cliccato con il tasto destro su " & Target.Address
    Cancel = True ' per disabilitare il menu contestuale
End Sub
```

Esempi pratici

Messaggio quando si modifica A1

```
If Target.Address = "$A$1" Then MsgBox "Hai modificato A1"
```

Bloccare modifiche a un intervallo specifico

```
If Not Intersect(Target, Range("B1:B5")) Is Nothing Then
    Application.Undo
    MsgBox "Modifica non consentita!"
```

Cambio colore della cella selezionata

```
Target.Interior.Color = RGB(200, 200, 255)
```

Disattivare il menu contestuale

```
Cancel = True nell'evento BeforeRightClick
```

Mostrare il contenuto della cella selezionata

```
MsgBox "Hai selezionato: " & Target.Value
```

Esercizi

Mostra un messaggio quando viene selezionata la cella C3

Usa Worksheet_SelectionChange e controlla Target.Address.

Blocca il clic destro sulla colonna D

Usa Worksheet_BeforeRightClick e controlla Target.Column = 4

Segnala modifiche in un intervallo preciso

Usa Intersect(Target, Range("E1:E10"))

Colora in rosso ogni cella modificata

```
Target.Interior.Color = RGB(255, 0, 0)
```

Aggiungi un timestamp in B1 ogni volta che A1 cambia

```
If Target.Address = "$A$1" Then Range("B1") = Now
```

6. Utilizzo di eventi speciali dell'applicazione (Application Events)

a. Utilizzo degli eventi speciali Application

Excel consente l'uso di eventi speciali legati all'applicazione, non limitati a fogli o cartelle. Tra questi, gli eventi Application.OnTime e Application.OnKey permettono un'interazione più flessibile e avanzata.

b. Utilizzo dell'evento OnTime

L'evento OnTime consente di pianificare l'esecuzione di una macro a un'ora specifica.

Sintassi:
Application.OnTime TimeValue("14:30:00"), "NomeMacro"

Può essere utile per operazioni automatiche programmate.

c. Utilizzo dell'evento OnKey

Permette di associare una combinazione di tasti a una macro.

Sintassi:
Application.OnKey "^q", "NomeMacro"

Nell'esempio, premendo Ctrl+Q verrà eseguita la macro 'NomeMacro'.

Esempi pratici

Eseguire una macro alle 12:00

```
Application.OnTime TimeValue("12:00:00"), "MostraMessaggio"
Sub MostraMessaggio()
    MsgBox "È mezzogiorno"
End Sub
```

Associare la combinazione Ctrl+M a una macro

```
Application.OnKey "^m", "MacroTest"
Sub MacroTest()
    MsgBox "Hai premuto Ctrl+M"
End Sub
```

Programmare un salvataggio automatico ogni ora

```
Application.OnTime Now + TimeValue("01:00:00"), "AutoSalva"
Sub AutoSalva()
    ThisWorkbook.Save
End Sub
```

Disattivare una scorciatoia

```
Application.OnKey "^q", ""  ' Rimuove l'associazione
```

Lanciare un promemoria programmato

```
Application.OnTime TimeValue("16:00:00"), "Promemoria"
Sub Promemoria()
    MsgBox "Fine giornata!"
End Sub
```

Esercizi

Scrivi una macro che venga eseguita automaticamente ogni 10 minuti
Usa Application.OnTime con Now + TimeValue("00:10:00")

Crea una macro associata a Ctrl+T che inserisca la data in A1

```
Range("A1").Value = Date
```

Programma una macro che salvi il file ogni 15 minuti
Usa Application.OnTime e ThisWorkbook.Save

Rimuovi un'associazione da tastiera esistente

```
Application.OnKey "^x", ""
```

Programma un messaggio motivazionale ogni ora

```
Sub Motivazione()
    MsgBox "Continua così!"
End Sub
Application.OnTime Now + TimeValue("01:00:00"), "Motivazione"
```