Oggetto Application – Metodi e Proprietà

L'oggetto *Application* di Excel rappresenta l'applicazione Microsoft Excel, definita anche applicazione Host. Se l'applicazione Host è Microsoft Word, l'oggetto Application si riferisce e rappresenta l'applicazione Word. Excel presuppone che anche quando non è specificato, il qualificatore dell'oggetto Application non è necessario per essere utilizzato nel codice VBA, perché l'applicazione di default è Excel stesso, a meno che non si vuole fare riferimento ad altre applicazioni esterne (come Microsoft Word o Access) nel codice o se si vuole fare riferimento a Excel da un'altra un'applicazione come Microsoft Word.

Tutte le applicazioni Microsoft Office, che utilizzano VBA, hanno un proprio modello di oggetti e durante la scrittura di codice VBA in Microsoft Excel, si prevede di utilizzare gli oggetti forniti dal modello a oggetti di Excel. Il modello a oggetti è una grande gerarchia di tutti gli oggetti utilizzati in VBA e il modello Application si riferisce e contiene i suoi oggetti di programmazione che sono legati gli uni agli altri in una gerarchia. L'intera applicazione Excel è rappresentata dall'oggetto Application che è in cima alla gerarchia di oggetti e scendendo verso il basso è possibile accedere agli oggetti per la cartella di lavoro, i fogli di lavoro e gli Intervalli di celle. Gli oggetti di Excel sono accessibili attraverso oggetti '*padri*', il foglio di lavoro è il genitore dell'oggetto Range, la cartella di lavoro è l'oggetto padre del foglio di lavoro, e l'oggetto Application è il padre dell'oggetto cartella di lavoro.

Brevemente si può dire che l'oggetto Application è la classe principale nel modello a oggetti e ogni volta che si apre Excel, viene istanziato un nuovo oggetto Application. La classe Application possiede metodi e proprietà come:

- *ActiveWorkBook*: proprietà che restituisce la cartella di lavoro attiva nell'applicazione
- *ActiveSheet*: proprietà che restituisce il foglio attivo nella cartella di lavoro attiva dell'applicazione
- *ActiveCell*: proprietà che restituisce la cella attiva nel foglio di lavoro attivo, della cartella di lavoro attiva dell'applicazione

Nel codice VBA, sia le espressioni *Application.ActiveWorkbook.Name* e *ActiveWorkbook.Name* avranno lo stesso effetto di restituire il nome della cartella di lavoro attiva. Tuttavia, ci sono alcuni casi in cui è richiesta la qualificazione dell'oggetto Application per essere utilizzato, vale a dire quando si utilizzano proprietà e metodi che riguardano l'aspetto della finestra di Excel, o che riguardano come l'applicazione Excel si deve comportare. Vediamo alcuni casi in cui il qualificatore Application deve essere utilizzato per proprietà o metodi dell'oggetto Application:

Proprietà Application.WindowState

La proprietà *WindowState* imposta lo stato della finestra dell'applicazione e le opzioni sono *xlNormal*, *xlMaximized* (imposta la finestra attiva alla dimensione massima disponibile purché non sia già ingrandita) e *xlMinimized*. Si noti che queste proprietà non possono essere impostate se la finestra è ingrandita, e le proprietà sono di sola lettura se la finestra è minimizzata

```
Sub app_winstat ()
   Application.WindowState = xlNormal
   Application.Height = 350
   Application.Width = 450
End Sub
```

Proprietà Application.DisplayFullScreen

Utilizza un valore Booleano e impostata su True, la finestra dell'applicazione è ingrandita alla dimensione massima e la barra del titolo dell'applicazione viene nascosta, sintassi:
Application.DisplayFullScreen = True

Proprietà Application.DisplayFormulaBar

Utilizza un valore Booleano e mostra o nasconde la barra della formula quando è impostato su, rispettivamente, True o False, sintassi: *Application.DisplayFormulaBar = False*

Proprietà Application.Calculation

Restituisce o imposta la modalità di calcolo e dispone di 3 impostazioni:

- *xlCalculationAutomatic* - (Default) ricalcola automaticamente i dati che vengono immessi nelle celle
- *xlCalculationSemiautomatic* - Ricalcolo automatico da Excel ad eccezione di tabelle di dati
- *xlCalculationManual* - Il calcolo viene fatto solo quando richiesto dall'utente facendo clic su 'Calcola ora' o premendo F9.

Esempio: *Application.Calculation = xlCalculationManual*

Proprietà Application.EditDirectlyInCell

Utilizza un valore booleano e consente o non consente la modifica direttamente nelle celle quando è impostata, rispettivamente a True o False, sintassi: *Application.EditDirectlyInCell = False*

Proprietà Application.ScreenUpdating

Utilizza un valore booleano ed è usata quando non si vuole vedere lo schermo seguire le azioni della routine VBA che si sta eseguendo. Se la proprietà *ScreenUpdating* è impostata su False, l'aggiornamento dello schermo non è visibile, e non sarà in grado di visualizzare ciò che il codice fa, rendendo più veloce l'esecuzione. Quando si disattiva l'aggiornamento dello schermo nelle procedure VBA al termine delle quali si deve impostare la proprietà a True, sintassi: *Application.ScreenUpdating = False*

Proprietà Application.DisplayAlerts

Utilizza un valore booleano e durante l'esecuzione di una macro, alcuni avvisi vengono visualizzati da Excel per confermare un'azione durante l'eliminazione di un foglio di lavoro. L'impostazione di questa proprietà su False non visualizzerà alcuna richiesta o avviso e in questo caso si riceverà una risposta predefinita da Excel. Questa proprietà deve essere impostata con il valore predefinito True dopo la procedura termina, sintassi: *Application.DisplayAlerts = False*

Proprietà Application.DefaultFilePath

Si usa questa proprietà per ottenere e impostare il percorso predefinito utilizzato da Microsoft Office Excel per caricare e salvare i file, sintassi: *Application.DefaultFilePath = "C:\Documenti\Excel"*

Metodo Application.Quit

Si utilizza questo metodo per chiudere l'applicazione Excel. Si noti che dopo la chiusura della cartella di lavoro, la finestra di Excel rimane aperta. Per uscire da Excel si può utilizzare il metodo Quit come illustrato di seguito. sintassi: *ThisWorkbook.Close SaveChanges: = True*

Chiude Excel e poi chiude ThisWorkbook dopo il salvataggio (il metodo Quit non termina Excel)
Application.Quit
ThisWorkbook.Close SaveChanges: = True

Chiude ThisWorkbook, ma non chiude Excel perché viene chiuso con ThisWorkbook senza leggere la linea Application.Quit:
ThisWorkbook.Close SaveChanges: = True
Application.Quit

Metodo Application.OnTime

Questo metodo viene utilizzato in VBA per eseguire automaticamente una procedura ad intervalli periodici o in un momento specifico della giornata. Nel seguente esempio, *RunTime* è una variabile pubblica di tipo Date, che imposta l'intervallo di tempo e la macro denominata MacroAutoRun verrà eseguito automaticamente, a intervalli di tempo programmato di tre secondi, con il metodo *OnTime,* sintassi: *Application.OnTime RunTime, "MacroAutoRun"*

Esempio di utilizzo del metodo Application.OnTime: Questa procedura utilizza il metodo OnTime col valore di incremento reperito in una cella a intervalli di tempo specifici, e Arresta la procedura dopo averla eseguita per un determinato numero di volte. Si Imposta l'intervallo di tempo 3 secondi, nel quale la procedura verrà eseguita

```
RunTime Public As Date
Dim count As Integer

Sub MacroAutoRun ()
RunTime = + TimeValue ("00:00:03") Now
Application.OnTime RunTime, "MacroAutoRun", True
 'si incrementa il valore nella cella A1 del foglio di lavoro di 5 unità, ogni volta che la macro si ripete
Cells (1, 1) Valore = Cells (1, 1).Value + 5
count = count + 1
'Interrompere la procedura dopo averla eseguita per 5 volte
If count = 5 Then
Application.OnTime RunTime, "MacroAutoRun", False
count = 0
End If
End Sub
```

Metodo Application.ActivateMicrosoftApp

Questo metodo attiva un'applicazione Microsoft già in esecuzione oppure crea una nuova istanza dell'applicazione nel caso in cui l'applicazione non è già in esecuzione. Qui di seguito i codici per avviare e attivare, rispettivamente, Word, Access e Power Point:

- *Application.ActivateMicrosoftApp xlMicrosoftWord*
- *Application.ActivateMicrosoftApp xlMicrosoftAccess*
- *Application.ActivateMicrosoftApp xlMicrosoftPowerPoint*

Metodo Application.GetOpenFilename

Il metodo *GetOpenFilename* ottiene il nome del file da aprire dall'utente, visualizzando la finestra di dialogo standard Apri. Il file non viene effettivamente aperto, ma restituisce il percorso completo e il nome del file selezionato. La sintassi è la seguente: *ApplicationObject.GetOpenFilename (FileFilter, FilterIndex, titolo, ButtonText, MultiSelect)*

Tutti gli argomenti sono opzionali, ma è necessario specificare l'oggetto Application.
L'argomento *FileFilter* è un valore stringa che specifica i criteri di filtro per il tipo di file che verranno visualizzati nella directory da cui l'utente ottiene il nome del file. Ci sono 4 filtri specificati in FileFilter ed è possibile specificare uno qualsiasi di questi criteri predefiniti utilizzando i numeri di indice da 1 a 4

Si può utilizzare l'argomento *Titolo* per specificare il titolo della finestra di dialogo. Il titolo predefinito è "Open" se non specificato. L'argomento *ButtonText* è applicabile solo per i computer che eseguono Excel per Macintosh (Mac) e impostando l'argomento *MultiSelect* a True si consente la selezione multipla dei file. L'impostazione predefinita è False. E' possibile utilizzare FileFilter per i file di Excel xlsm, specificando le estensioni in coppie e utilizzando caratteri jolly *"Excel Files (*. Xlsm), *. Xlsm"*

Si può usare FileFilter per estensioni xls, xlsx e XLSM, usando il punto e virgola e i caratteri jolly *"Excel Files (* xls,. *. Xlsx,. * Xlsm), * xls, *. Xlsx,. *. Xlsm"* Utilizzando FileFilter per estensioni XLSM e file di testo txt elencandoli separatamente nella lista dei tipi di file *"Excel Files (*. Xlsm), *. Xlsm, File di testo (*. Txt), *. Txt"*
Se l'argomento FileFilter viene omesso, per impostazione predefinita assume la sintassi: tutti i file *"Tutti i file (*. *), *. *"*

Esempio: Selezionare e aprire un singolo file, immettere il salvare e chiudere la cartella di lavoro, utilizzando i metodi Application.GetOpenFilename e Workbooks.Open

```
Sub GetOpenFilename1 ()
  Dim nome_file As Variant
  Dim wkbk As Workbook
  Dim str_cartella As String
  Dim cor_cartella As String
cor_cartella = CurDir
str_cartella = "C:\Test"
ChDrive str_cartella 'impostata l'unità corrente
ChDir str_cartella
nome_file = Application.GetOpennome_file(FileFilter:="Excel .. Files (* xlsm), * xlsm ",
Title:="Seleziona un file ")
If nome_file = False Then
MsgBox "Selezionare un file per continuare"
Exit Sub
Else
Workbooks.Open (nome_file)
Set wkbk = ActiveWorkbook
Sheets("Foglio1").Select
wkbk.ActiveSheet.Range("A1") = "Ciao"
 wkbk.Save
wkbk.Close
End If
ChDrive cor_cartella
ChDir cor_cartella
End Sub
```

Esempio: Selezionare e aprire più file, utilizzando i metodi Application.GetOpenFilename e Workbooks.Open

```
Sub GetOpenFilename2 ()
 Dim nome_file As Variant
 Dim i As Integer
 Dim iFiles As Integer
'impostando MultiSelect a True il metodo GetOpenFilename consente la selezione di file multipli
nome_file = Application.GetOpenFilename(filefilter:="Excel Files(*.xls;*.xlsx;*.xlsm),
*.xls;*.xlsx;*.xlsm", Title:="Seleziona file", MultiSelect:=True)
'Verifica se l'utente fa clic sul pulsante Annulla
If IsArray(nome_file) = False Then
MsgBox "Seleziona il file per continuare"
Exit Sub
Else
iFiles = UBound(nome_file) - LBound(nome_file) + 1 'Determinare il numero di file da aprire
For i = 1 To iFiles
Workbooks.Open nome_file(i) 'Metodo Workbooks.Open apre una cartella di lavoro
Next i
End If
End Sub
```

Esempio: Aprire un file Excel esistente, aggiungere una nuova cartella nella stessa posizione e salvarlo come un file xlsm, poi copiare un foglio dalla cartella di lavoro esistente alla cartella di lavoro appena aggiunta, quindi chiudere entrambe le cartelle di lavoro.

```
Sub GetOpenFilename3()
Dim fileName As Variant
Dim str_cart As String
Dim wb_dest As Workbook
Dim wb_orig As Workbook
str_cart = "C:\Test" 'impostare un percorso da cui selezionare i file nella finestra di dialogo Apri
ChDrive str_cart 'impostare l'unità
ChDir str_cart 'impostare la directory corrente
fileName = Application.GetOpenFilename(FileFilter:="Excel .. Files (* xlsm), * xlsm ",
Title:="Seleziona un file ")
If fileName = False Then 'Se l'utente fa clic sul pulsante Annulla viene visualizzato un avviso a video
MsgBox "Si prega di selezionare un file per continuare"
Exit Sub
Else
Set wb_orig = Workbooks.Open(fileName) 'apre una cartella di lavoro
End If
Workbooks.Add 'Aggiungere una nuova cartella di lavoro, che è la cartella di lavoro di destinazione
'salva la cartella di lavoro di destinazione con un nuovo nome e il tipo di file xlsm
ActiveWorkbook.SaveAs fileName:="newWorkbook.xlsm", FileFormat:=52
Set wb_dest = ActiveWorkbook
'copia il foglio2 dalla cartella di origine alla nuova cartella di destinazione come ultimo foglio
wb_orig.Worksheets("Foglio2").Copy After:=wb_dest.Sheets(Sheets.Count)
'chiude sia le cartelle di lavoro, dopo aver salvato la cartella di destinazione
wb_orig.Close
wb_dest.Close SaveChanges:=True
End Sub
```

Proprietà e metodi dell'oggetto Application

Tali proprietà e metodi il cui utilizzo **NON** richiede di specificare il qualificatore dell'oggetto Application sono considerati *globali*. È possibile visualizzare queste proprietà e metodi globali nel Visualizzatore oggetti in VBE dal menu **Visualizza - Visualizzatore oggetti**, o premendo F2 e scegliendo Excel dall'elenco a discesa delle librerie nel riquadro superiore e quindi scegliendo *GLOBALS* che appare in alto nella casella Classi. I Casi in cui non è richiesta l'utilizzazione del qualificatore dell'applicazione sono:

Proprietà ActiveCell

La proprietà *ActiveCell*, applicata ad un oggetto Application, restituisce la cella attiva (oggetto Range) del foglio di lavoro visualizzato nella finestra attiva. È inoltre possibile applicare questa proprietà per un oggetto *Window* specificando la finestra per cercare la cella attiva. Si noti che, in assenza di un foglio di lavoro visualizzato nella finestra, la proprietà avrà esito negativo. Qualsiasi dei seguenti codici possono essere utilizzati in alternativa (si noti che Value è la proprietà predefinita di un oggetto Range)

MsgBox "il valore della cella attiva è:" & Application.ActiveCell
MsgBox "il valore della cella attiva è:" & ActiveCell
MsgBox "il valore della cella attiva è:" & ActiveWindow.ActiveCell
MsgBox "il valore della cella attiva è:" & ActiveCell.Value

Proprietà ActiveWindow

Questa proprietà restituisce la finestra attiva. La finestra attiva è la finestra attualmente selezionata. Il codice per visualizzare il nome del'[i]ActiveWindow[i] che appare nella barra del titolo è il seguente:

MsgBox "La finestra attiva è:" & Application.ActiveWindow.Caption
MsgBox "La finestra attiva è:" & ActiveWindow.Caption

Proprietà ActiveWorkbook

Questa proprietà restituisce la cartella di lavoro attiva, cioè la cartella di lavoro nella finestra attiva

MsgBox "il nome della cartella attiva è" & Application.ActiveWorkbook.Name
MsgBox "il nome della cartella attiva è" & ActiveWorkbook.Name

Proprietà ThisWorkbook

Questa proprietà viene utilizzata solo dall'interno dell'applicazione Excel e restituisce la cartella di lavoro in cui il codice viene eseguito al momento.

MsgBox "Il nome di questa cartella di lavoro è" & Application.ThisWorkbook.Name
MsgBox "Il nome di questa cartella di lavoro è" & ThisWorkbook.Name

Si noti che sebbene il più delle volte ActiveWorkbook è la stessa ThisWorkbook, ma potrebbe non essere sempre così. La cartella di lavoro attiva può essere diversa da quella di lavoro in cui viene eseguito il codice, come illustrato dal seguente esempio.

```
Sub ActiveWorkbook_ThisWorkbook ()
MsgBox "il nome della cartella Attiva è" & ActiveWorkbook.Name
MsgBox "Il nome di questa cartella di lavoro è" & ThisWorkbook.Name
Workbooks ("Book2. xlsm "). Activate
MsgBox "il nome della cartella Attiva è" & ActiveWorkbook.Name
MsgBox "Il nome di questa cartella di lavoro è" & ThisWorkbook.Name
Workbooks ("Book1.xlsm"). Activate
MsgBox "il nome della cartella Attiva è" & ActiveWorkbook.Name
MsgBox "Il nome di questa cartella di lavoro è" & ThisWorkbook.Name
End Sub
```

Proprietà ActiveSheet

La proprietà *ActiveSheet*, applicata ad un oggetto Application, restituisce il foglio attivo nella cartella di lavoro attiva. È inoltre possibile applicare questa proprietà per una cartella di lavoro o finestra specificando la cartella di lavoro o la finestra per cercare il foglio attivo. I seguenti codici mostrano il foglio attivo nella cartella di lavoro attiva

Msgbox "il nome del foglio attivo è" & Application.ActiveSheet.Name
MsgBox "il nome del foglio attivo è" & Application.ActiveWorkbook.ActiveSheet.Name
Msgbox "il nome del foglio attivo è" & ActiveSheet.Name
MsgBox "il nome del foglio attivo è" & ActiveWorkbook.ActiveSheet.Name

Mentre il seguente codice restituisce il foglio attivo nella cartella di lavoro specificata (denominata "Test_vba.xlsm"):

MsgBox "il nome del foglio attivo è" & Application.Workbooks ("Test_vba.xlsm"). ActiveSheet.Name

Proprietà ActiveChart

La proprietà *ActiveChart*, applicata ad un oggetto Application, restituisce il grafico attivo nella cartella di lavoro attiva. Nella cartella di lavoro è possibile avere fogli grafici separati e definisce grafici come fogli di lavoro o grafici incorporati che comprende il grafico come un oggetto all'interno di un foglio di lavoro. Un foglio grafico o un grafico incorporato è attivo se è stato selezionato o se è attivato con il metodo Activate. È inoltre possibile applicare la proprietà ActiveChart di una cartella di lavoro o un oggetto Window specificando la cartella di lavoro o la finestra per cercare il grafico attivo. I seguenti codici mostrano il grafico attivo nella cartella di lavoro attiva

MsgBox "Il nome del grafico attivo è:" & Application.ActiveChart.Name
MsgBox " Il nome del grafico attivo è:" & Application.ActiveWorkbook.ActiveChart.Name
MsgBox " Il nome del grafico attivo è:" & ActiveChart.Name
MsgBox " Il nome del grafico attivo è:" & ActiveWorkbook.ActiveChart.Name

Il seguente codice restituisce il grafico attivo nella cartella di lavoro specificata (denominata " Test_vba.xlsm ")

MsgBox " Il nome del grafico attivo è:" & Application.Workbooks ("Test_vba.xlsm ") ActiveChart.Name

Proprietà Application.ActivePrinter

Questa proprietà imposta il nome della stampante attiva. Per Impostare una stampante HP LaserJet 0011 su LPT1 come stampante attiva

Application.ActivePrinter = "HP LaserJet 0011 su NE02: "
ActivePrinter = "HP LaserJet 0011 su NE02: "

Per impostare la stampante Adobe PDF sul Ne04 come stampante attiva

Application.ActivePrinter = "Adobe PDF su Ne04: "
ActivePrinter = "Adobe PDF su Ne04: "

Proprietà Selection

La proprietà *Selection* se applicata a un oggetto Application, restituisce l'oggetto che viene selezionato nella finestra attiva. L'oggetto selezionato potrebbe essere un oggetto Range o una singola cella del foglio di lavoro attivo, un oggetto ChartArea nel foglio di lavoro attivo, e così via. È inoltre possibile applicare la proprietà di selezione un oggetto finestra in cui la proprietà restituirà l'oggetto selezionato nella finestra specificata. Presumendo che la selezione corrente sia un oggetto Range nel foglio di lavoro attivo.

Per cancellare il contenuto delle celle selezionate nel foglio di lavoro attivo, si può utilizzare uno dei due metodi sotto descritti con o senza il qualificatore di oggetto Application

Application.Selection.Clear
Selection.Clear

Esempio: Selezionare un intervallo di celle per ottenere il tipo di oggetto della selezione e utilizzare la selezione per cambiare colore e grandezza carattere.

```
Sub SelectionProperty ()
Range ("A1: C3"). Select
'ottenere il tipo di oggetto della selezione - restituisce "Range"
MsgBox TypeName (Selection)
'cambiare il colore del carattere del campo selezionato
Selection.Font.Color = vbRed
'cambiare la dimensione del carattere dell'intervallo selezionato
Selection.Font.Size = 14
End Sub
```

Proprietà Fogli

L'oggetto *Sheets* si riferisce a tutti i fogli contenuti in una cartella di lavoro, che comprende fogli grafici e fogli di lavoro. Le Schede della struttura, applicate ad un oggetto Application, restituisce un insieme Sheets nella cartella di lavoro attiva. È inoltre possibile applicare questa proprietà per una cartella di lavoro specificando la cartella di lavoro che restituirà una raccolta di fogli di lavoro specificato. Qui di seguito viene illustrato come contare il numero di fogli della cartella di lavoro attiva e restituire il nome di ciascun foglio.

```
Sub SheetsCollection ()
Dim n As Integer
'numero dei fogli nella cartella di lavoro attiva
MsgBox " Numero di fogli nella cartella di lavoro attiva sono:" & Application.Sheets.Count
MsgBox " Numero di fogli nella cartella di lavoro attiva sono:" & Sheets.Count
'Nome dei fogli in base al valore dell'indice
For n = 1 To Sheets.Count
MsgBox Sheets(n). Name
Next n
'Nome dei fogli scorrendoli tutti nella cartella di lavoro
For Each Sheet In Sheets
MsgBox Sheet.Name
Next
End Sub
```

Proprietà Intervallo

La proprietà *Range* restituisce un oggetto Range (singola cella o intervallo di celle). È possibile utilizzare la sintassi: *Range (cell1)*. Questo può essere solo un riferimento di tipo A1, ed è possibile utilizzare un operatore di intervallo o l'operatore di unione (es. virgola), oppure può essere un intervallo denominato. È inoltre possibile utilizzare la sintassi: *Range (cell1, cell2)* - dove cell1 e cell2 sono oggetti che rappresentano l'intervallo di celle contigue che specificano le celle di inizio e fine. Per applicare la proprietà Range di un oggetto Application, è possibile omettere il qualificatore di oggetto. Range (cell1) o utilizzare *Application.Range (cell1)*, entrambi restituiranno l'oggetto Range in ActiveSheet. Per applicare la proprietà Range di un oggetto cartella di lavoro o un oggetto Range, specificare il rispettivo qualificatore oggetto come illustrato di seguito.

Proprietà Range applicabili all'oggetto Application

Qualsiasi dei seguenti codici inseriranno il valore 10 nella cella A1 del foglio attivo (con o senza usare il qualificatore di oggetto Application o digitando ActiveSheet)

Application.ActiveSheet.Range ("A1"). Value = 10
ActiveSheet.Range ("A1"). Value = 10
Application.Range ("A1"). Value = 10
Range ("A1"). Value = 10

Il seguente codice inserirà il valore 10 nelle celle A1, A2, A3 e A4 del foglio attivo. La seconda espressione utilizza la sintassi: Range (cell1, cell2):

Range ("A1: A4") Valore = 10
Range ("A1", "A4"). Value = 10

Il seguente codice inserirà il valore 10 nelle celle A1, B2 e C4 del foglio attivo (si noti che specificando "Value" dopo il Range non è necessario perché è la proprietà predefinita dell'oggetto Range)

Range ("A1, B2, C4") = 10

Il seguente codice inserirà il valore 10 nell'intervallo denominato "pippo" del foglio attivo, vale a dire che è possibile denominare l'intervallo A1 come "pippo" e utilizzare il seguente codice:

Range ("pippo") = 10

Proprietà Range applicabili all'oggetto Worksheet

Il seguente codice inserirà il testo "Carlo" nella cella A1 del foglio di lavoro denominato "Foglio1": *Worksheets ("Foglio1"). Range ("A1") = "Carlo"*

Proprietà Range applicabili all'oggetto Range

Quando la proprietà Range viene applicata a un oggetto Range, la proprietà diventa relativa all'oggetto Range come illustrato di seguito:

```
Sub RangeProperty ()
'Selezionare un intervallo nel foglio attivo
Range("C5:E8").Select
'inserisce il valore 10 in C5
Selection.Range("A1") = 10
'inserisce il valore 11 in C6
Selection. Range ("A2") = 11
'inserisce il valore 20 in D5
Selection.Range ("B1") = 20
'inserisce il valore 20 in D6
Selection.Range ("B2") = 21
End Sub
```

Metodo Calcola

Il metodo Calcola, se applicato a un oggetto Application, calcola tutte le cartelle di lavoro aperte. È inoltre possibile applicare questo metodo a un oggetto foglio di lavoro specificando il foglio di lavoro in una cartella di lavoro; oppure è possibile applicare questo metodo a un intervallo specificando una cella o un intervallo di celle in un foglio di lavoro.

Metodo Calcola applicabile all'oggetto Application
Utilizzare una delle seguenti righe di codice per calcolare tutte le cartelle di lavoro aperte:
Application.Calculate
Calculate

Metodo Calcola applicabile all'oggetto Worksheet
Utilizzare una delle seguenti righe di codice per calcolare un foglio di lavoro specifico ("Foglio1"):
Application.Worksheets ("Foglio1"). Calculate
Worksheets ("Foglio1"). Calculate

Metodo Calcola applicabile all'oggetto Range
Utilizzare una delle seguenti righe di codice per calcolare l'intervallo specificato (celle A5, A6 e A7) in un foglio di lavoro:
Application.Worksheets ("Sheet1") Range ("A5: A7"). Calculate.
Worksheets ("Foglio1") Range ("A5: A7"). Calculate

La seguente riga di codice calcola l'intera colonna (colonna A) in un foglio di lavoro:
Worksheets ("Foglio1"). Columns (1). Calculate

La seguente riga di codice calcola le celle specifiche A5, A6, B7 e B20 in un foglio di lavoro:
Worksheets ("Foglio1"). Range ("A5, A6, B7, B20"). Calculate

Utilizzare la seguente riga di codice per calcolare l'intervallo specificato (celle A5, A6 e A7) nel foglio di lavoro attivo: *Range ("A5: A7"). Calculate*

Velocizzare il codice VBA disattivando gli aggiornamenti dello schermo e i calcoli automatici
In Excel, la modalità di calcolo predefinita è la Modalità Automatica. sintassi: (*Application.Calculation = xlCalculationAutomatic*), in cui viene calcolata automaticamente ogni cella quando si entra. Quando Excel è in modalità manuale: (*Application.Calculation = xlCalculationManual*)

il calcolo viene effettuato solo quando richiesto dall'utente facendo clic su "*Calcola Ora*" o premendo F9 oppure cambiando la modalità di calcolo automatico. In modalità automatica, per ciascun nuovo valore immesso da una macro, Excel ricalcolerà tutte le celle che verranno coinvolte dal nuovo valore rallentando così l'esecuzione del codice VBA. Questo può rallentare notevolmente il funzionamento del codice vba, soprattutto nel caso di grandi macro con moli di calcolo significativi. Per accelerare una macro e rendere l'esecuzione più veloce ed efficiente, è tipico degli sviluppatori disattivare il calcolo automatico all'inizio della macro, ricalcolare il foglio di lavoro specifico o l'intervallo di celle utilizzando il metodo Calculate all'interno della macro e poi riportare il calcolo in automatico alla fine del codice.

L'esempio seguente disattiva gli aggiornamenti dello schermo e i calcoli automatici e utilizza il metodo Calculate, durante l'esecuzione del codice

```
Sub CalculateMethod ()
Application.ScreenUpdating = False 'disattivare Aggiornamenti schermo e calcoli automatici
Application.Calculation = xlCalculationManual
….. altro codice
Application.Calculate 'Utilizzare il metodo Calculate per calcolare tutte le cartelle di lavoro aperte
Application.ScreenUpdating = True 'attivare gli aggiornamenti dello schermo e calcoli automatici
Application.Calculation = xlCalculationAutomatic
End Sub
```