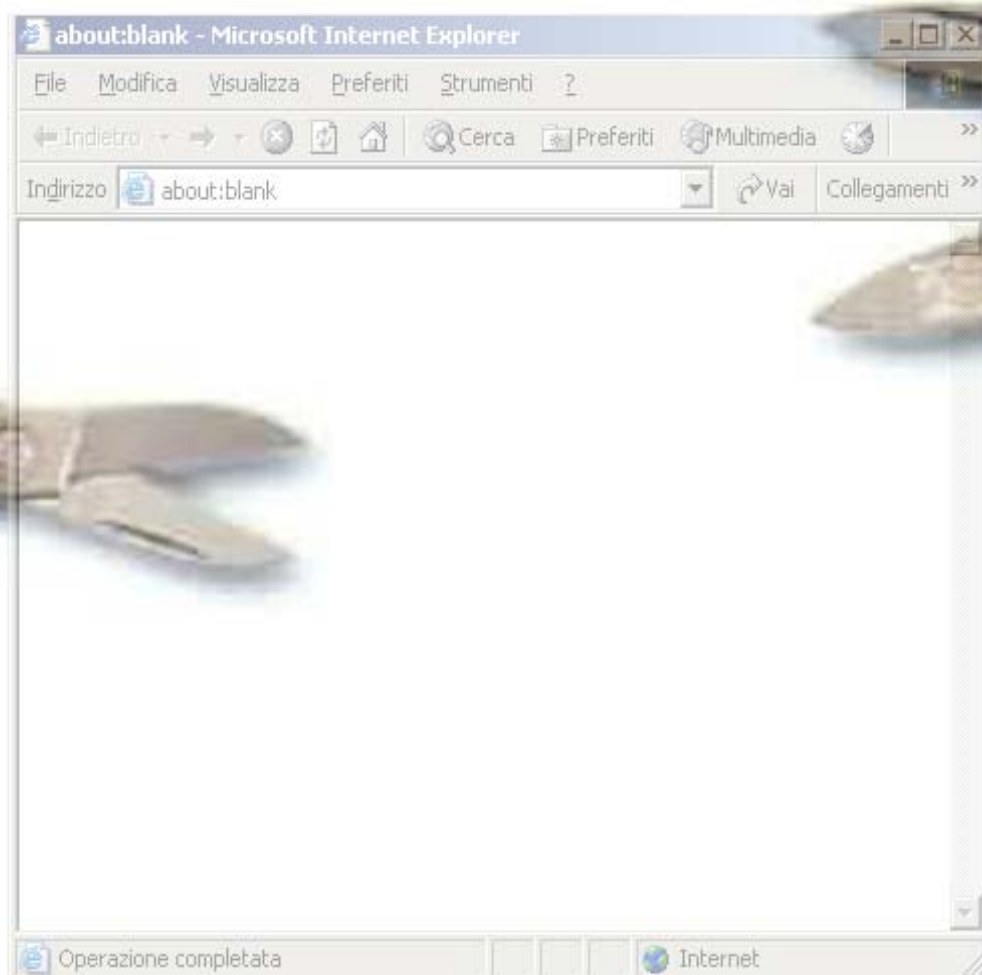


VBSCRIPT: GUIDA DI RIFERIMENTO

***A cura di
Paolo Malesci***



Visual Basic Italia

VBScript: Guida di riferimento

A cura di Paolo Malesci

Indice

Introduzione.....	3
Le variabili.....	5
Gli Operatori.....	6
Istruzioni di Dichiarazione.....	7
Istruzioni di controllo del flusso.....	9
Altre istruzioni.....	11
Funzioni implicite.....	12
Gli Oggetti.....	16

Introduzione

VbScript è un linguaggio di *scripting* della Microsoft derivato dal linguaggio di programmazione Visual Basic e, come quest'ultimo, è un linguaggio orientato agli oggetti (*object oriented*).

Tra i vari linguaggi di *scripting* è quello più *english-like* e ciò ne spiega la larga diffusione.

Ogni istruzione è costituita da una riga, termina quindi con il ritorno a capo; se risulta troppo lunga per la leggibilità può essere continuata su una (o più) riga successiva utilizzando a fine riga il carattere `_` (*underslash*) per indicare la continuazione sulla riga successiva.

Per le istruzioni possono essere usate anche tutte lettere minuscole, anche se noi ci atterremo alla forma standard.

Il carattere di delimitazione delle stringhe è `"` (*doppio apice*).

1. [Le Variabili](#)
2. [Gli Operatori](#)
3. [Istruzioni di Dichiarazione](#)
4. [Istruzioni di controllo del flusso](#)
5. [Altre istruzioni](#)
6. [Funzioni implicite](#)
7. [Gli Oggetti](#)

1 – Le Variabili

I nomi delle variabili devono sempre iniziare con una lettera (maiuscola o minuscola) e possono contenere cifre ed il carattere `_`; non possono contenere né spazi né altri caratteri speciali.

Sono con visibilità **Public** le variabili dichiarate a livello dello Script, con visibilità **Private** le variabili dichiarate all'interno di una Funzione o di una Subroutine; quest'ultime esistono quindi solo nell'ambito della Funzione o della Subroutine nella quale vengono dichiarate.

Qualora lo script fosse molto complesso è consigliabile attenersi alle raccomandazioni di Microsoft sulle convenzioni da seguire per i nomi delle variabili; consiste nell'uso di prefissi per indicare la visibilità ed il sottotipo.

s = dichiarata a livello di Script (Public)

nessuno = dichiarata a livello di Function o Sub (Private)

segue un prefisso di tre lettere per il sottotipo:

Sottotipo	Prefisso	Esempio
Boolean	bln	blnFound
Byte	byt	
Date	dtm	dtmOggi
Double	dbl	dblImporto
Error	err	
Integer	int	intQuantità
Long	lng	
Object	obj	
Single	sng	sngSconto
String	str	strNome

Io personalmente preferisco farne a meno, ma a volte può essere utile.

Esiste una analoga tabella di prefissi raccomandati per gli Oggetti:

Object type	Prefisso	Esempio
3D Panel	pnl	pnlGroup
Animated button	ani	aniMailBox
Check box	chk	chkReadOnly
Combo box, drop-down list box	cbo	cboEnglish
Command button	cmd	cmdExit
Common dialog	dlg	dlgFileOpen
Frame	fra	fraLanguage
Horizontal scroll bar	hsb	hsbVolume
Image	img	imgIcon
Label	lbl	lblHelpMessage
Line	lin	linVertical
List Box	lst	lstPolicyCodes
Spin	spn	spnPages
Text box	txt	txtLastName
Vertical scroll bar	vsb	vsbRate
Slider	sld	sldScale
Object generico	obj	

Questa può essere più utile per la leggibilità del programma.

2 – Gli Operatori

Possono essere aritmetici, di comparazione o logici.

Gli operatori sono in genere preceduti e seguiti da spazio, tuttavia l'interprete VbScript è in grado di riconoscerli anche in assenza degli spazi qualora non vi sia ambiguità.

Operatori Aritmetici

+	addizione
-	sottrazione
*	moltiplicazione
^	elevazione a potenza
/	divisione
\	divisione tra interi
Mod	modulo (resto della divisione di interi)
&	concatenazione (di stringhe)

Operatori di Comparazione

=	uguale
<>	diverso
<	minore
<=	minore o uguale
>=	maggiore o uguale
>	maggiore
Is	confronto di 2 variabili che fanno riferimento ad oggetti; = True se è lo stesso oggetto

Operatori Logici

And	intersezione
Or	unione
Not	negazione
Xor	Or esclusivo (<i>l'uno o l'altro ma non entrambi</i>)
Eqv	equivalenza logica; se espressioni numeriche uguaglianza a bit
Imp	implicazione logica; se espressioni numeriche bit espr2 >= bit espr1

3 – Istruzioni di Dichiarazione

- **Dim**
- **Redim**
- **Function / End Function**
- **Sub / End Sub**

Dim

Sintassi: **Dim** *variabile*[(*indici*)]

è l'istruzione di dichiarazione delle variabili: le variabili sono considerate *Public* se dichiarate a livello Script, *Private* se dichiarate a livello di Function o Sub; è possibile dichiarare Matrici indicando il valore massimo degli indici (il primo elemento ha indice=0):

`Dim Tabella(3, 5)` definisce una matrice 4 x 6.

Le variabili sono sempre di tipo *Variant* e non esistono quindi le clausole **As *tipovariabile***.

E' possibile la dichiarazione implicita, ovvero l'utilizzo di una istruzione di assegnazione ad una variabile non dichiarata:

```
SubTotale = 0
```

definisce la variabile SubTotale e le assegna il valore 0.

E' possibile rendere obbligatoria la dichiarazione delle variabili con la clausola (da mettere prima di tutte le dichiarazioni) **Option explicit**.

Non esistendo una istruzione specifica di dichiarazione delle costanti per i nomi di quest'ultime Microsoft consiglia l'utilizzo delle lettere maiuscole, ad es: `USER_LIST_MAX`, in modo da poterle sempre riconoscere ed evitare quindi di modificarne il valore per disattenzione.

Redim

Sintassi: **Redim** *variabile*(*nuoviindici*) [**Preserve**]

ridimensiona la variabile ai nuovi indici; se indicata la clausola *Preserve* il contenuto degli elementi della precedente matrice viene conservato.

Esempio:

```
Dim Nomi(4)
Nomi(0) = "Pippo"
Nomi(1) = "Pluto"
Nomi(2) = "Paperino"
Nomi(3) = "Qui"
Nomi(4) = "Quo"
.....
.....
Redim Nomi(5) Preserve
Nomi(5) = "Qua"
```

Function / End Function

Sintassi:

Function Nomefunzione(*elenco variabili*)

[. . istruzioni]

[Nomefunzione = espressione]

[Exit Function]

[. . istruzioni]

[Nomefunzione = espressione]

End Function

Definisce una funzione che restituisce o meno un valore effettuando elaborazioni sugli argomenti passati.

Esempio:

```

Function SDelta(a,b,c)
Dim d
    d = b * b + 4 * a + c
    If d >= 0 Then
        SDelta = Sqr(d)
    End If
End Function

```

da notare che se $b^2 - 4ac < 0$ ad SDelta non viene attribuito alcun valore.

Sub / End Sub

Sintassi:

```

Sub Nomeroutine(elenco variabili)
[. . istruzioni]
[Exit Sub]
[. . istruzioni]
End Sub

```

Definisce una routine che effettua elaborazioni sugli argomenti passati ed eventualmente su variabili Public.

Esempio:

```

Sub Ordina(a,b)
Dim d
    If a > b Then
        d = b
        b = a
        a = d
    End If
End Sub

```


4 – Istruzioni di controllo del Flusso

- **If .. Then**
- **Select Case**
- **Do / Loop**
- **While / Wend**

If ... Then

Sintassi:

If *condizione* **Then** *istruzione1* [**Else** *istruzione2*]

Oppure:

```
If condizione Then  
    istruzioni1  
    [ElseIf condizione2 Then  
        istruzioni2]  
    [Else  
        istruzioni3]
```

End If

Nel 2° formato (su più linee) *istruzioni1*, *istruzioni2*, *istruzioni3* possono essere anche di più linee. Si possono avere anche più blocchi **ElseIf**, ma non possono essere successivi alla clausola **Else**. E' inoltre possibile nidificare l'istruzione **If**; ad esempio:

```
If Numero = 1 Then  
    . . .  
Else  
    If Numero = 2 Then  
        . . .  
    Else  
        . . . ` diverso da 1 e 2  
    End If  
End If
```

Select Case

Sintassi:

```
Select Case espressione  
    Case espressione-1  
        istruzioni-1  
    [Case espressione-n  
        istruzioni-n]  
    [Case Else  
        istruzioni-else]
```

End Select

espressione può essere sia una variabile che qualsiasi espressione valida numerica o testo.

Viene eseguito il solo blocco di istruzioni corrispondente al caso trovato vero; se nessuno dei casi è vero viene eseguito l'eventuale blocco **Else**.

Consente di eseguire dei test multipli senza ricorrere a complicate nidificazioni di **If .. Then**.

For / Next

Sintassi:

```
For variabile = intero1 To intero2 [Step intero3]  
    istruzioni
```

Next

è supportata l'istruzione **Exit For**.

Al termine del ciclo il valore della variabile è intero2 + intero3: da tenere presente qualora si sia usata Exit For e si voglia controllare se il ciclo è stato completato:

```
For I = 0 To UBound(Nomi)    (non avendolo indicato Step = +1)
    If Utente = Nomi(I) Then Exit For
Next
If I > UBound(Nomi) Then
    Trovato = False
Else
    Trovato = True
End If
```

Do / Loop

Sintassi:

Do while|until *condizione*
 istruzioni

Loop

oppure

Do

istruzioni

Loop while|until *condizione*

è supportata l'istruzione **Exit Do**.

Esegue ciclicamente le istruzioni fintanto (while) condizione è vera o finchè (until) condizione non diventa vera; nel 1° caso il controllo viene effettuato all'inizio del ciclo mentre nel 2° viene effettuato dopo la prima esecuzione del blocco di istruzioni.

While / Wend

Sintassi:

While *condizione*
 istruzioni

Wend

Esegue ciclicamente le istruzioni fintanto la condizione è vera.

5 – Altre Istruzioni

- **Set**
- **Call**
- **Erase**
- **On Error**
- **Randomize**

Set

Sintassi:

Set *variabileoggetto* = [*espressioneoggetto* | **Nothing**]

Assegna alla variabile il riferimento all'oggetto specificato; *espressioneoggetto* può essere sia il nome di un oggetto sia una funzione od espressione che restituisce un oggetto.

L'assegnazione della variabile a **Nothing** interrompe il riferimento all'oggetto liberando le risorse allocate.

Call

Sintassi:

[**Call**] *nomeprocedura* [(*lista argomenti*)]

Esegue la procedura (*Sub* o *Function*) specificata; se **Call** è omissso la lista degli argomenti non deve essere racchiusa tra parentesi.

Se la procedura modifica argomenti è necessario indicare la clausola **ByVal** prima dell'argomento modificabile; inoltre se la procedura è una *Function* che restituisce un valore il valore di ritorno non viene passato.

Erase

Sintassi:

Erase *Matrice*

Reinizializza una matrice a dimensioni fisse o dealloca la memoria per le matrici dinamiche.

Dopo la sua esecuzione per le matrici dinamiche è necessaria l'istruzione **Redim** prima di poterle utilizzare nuovamente.

On Error

Sintassi:

On Error Resume Next

Abilita una routine di gestione degli errori.

Randomize

Sintassi:

Randomize [*numero*]

Inizializza il generatore di numeri casuali usando *numero* come origine per l'algoritmo; se è omissso viene usato il valore del timer di sistema.

Per ripetere una sequenza di numeri casuali non è sufficiente usare lo stesso valore di *numero*; occorre chiamare **Rnd** con un argomento negativo immediatamente prima di **Randomize**.

6 – Funzioni implicite

Non utilizzare mai il \$ dopo il nome della funzione (esempio: Chr\$ invece di Chr).

Categorie di Funzioni:

- [Matematiche](#)
- [Manipolazione Date](#)
- [Manipolazione Stringhe](#)
- [di Formattazione](#)
- [di Conversione](#)
- [Logiche](#)

Funzioni Matematiche

Abs (<i>numero</i>)	Valore assoluto
Atn (<i>numero</i>)	Arcotangente - <i>numero in radianti</i>
Cos (<i>numero</i>)	Coseno - <i>numero in radianti</i>
Exp (<i>numero</i>)	Esponenziale e^{numero}
Fix (<i>numero</i>)	Parte intera
Int (<i>numero</i>)	Valore intero arrotondato
LBound (<i>matrice</i> , [<i>dimensione</i>])	Valore minimo dell'indice
Log (<i>numero</i>)	Logaritmo naturale
Rnd [<i>(numero)</i>]	Numero casuale
Sgn (<i>numero</i>)	Segno - -1,+1
Sin (<i>numero</i>)	Seno - <i>numero in radianti</i>
Sqr (<i>numero</i>)	Radice quadrata
Tan (<i>numero</i>)	Tangente - <i>numero in radianti</i>
UBound (<i>matrice</i> , [<i>dimensione</i>])	Valore massimo dell'indice

Funzioni Manipolazione Date

	Tipo restituito	
Date	Date	Data del sistema
DateSerial (<i>anno, mese, giorno</i>)	Date	
DateValue (<i>stringa data</i>)	Date	
Day (<i>date</i>)	Numerico 1-31	
Hour (<i>date</i>)	Numerico 0-23	
Minute (<i>date</i>)	Numerico 0-59	
Month (<i>date</i>)	Numerico 1-12	
Now	DateTime	Data/Tempo del sistema
Second (<i>date</i>)	Numerico	
Time	DateTime < 1	Tempo del sistema
TimeSerial (<i>ore, minuti, secondi</i>)	DateTime < 1	
TimeValue (<i>stringa ora</i>)	DateTime < 1	
WeekDay (<i>date</i> [, <i>firstdayofweek</i>])	1-7 (1=Domenica)	
Year (<i>date</i>)	Numerico	

Attenzione nell'utilizzo di DateValue e TimeValue; il formato delle stringhe deve essere quello delle impostazioni internazionali del Computer su cui viene eseguito lo script.

Analogamente è meglio in WeekDay specificare *firstdayofweek* (generalmente è 1) se si ignorano le impostazioni del Computer.

Funzioni Manipolazione Stringhe

Asc (<i>carattere</i>)	Codice ASCII (0-255) del carattere
Chr (<i>codice carattere</i>)	Carattere corrispondente
InStr ([<i>start</i> ,] <i>stringacercata</i> , <i>stringa</i> [, <i>compare</i>])	Restituisce la posizione di <i>stringacercata</i> all'interno di <i>stringa</i> , se non trovata = 0; <i>start</i> default = 1
LCase (<i>stringa</i>)	Converte in minuscolo
Left (<i>stringa</i> , <i>lunghezza</i>)	Estrae a sinistra i caratteri indicati
Len (<i>stringa</i>)	Lunghezza della stringa
LTrim (<i>stringa</i>)	Elimina blank iniziali
Mid (<i>stringa</i> , <i>start</i> [, <i>lunghezza</i>])	Estrae i caratteri indicati a partire da <i>start</i>
Right (<i>stringa</i> , <i>lunghezza</i>)	Estrae a destra i caratteri indicati
RTrim (<i>stringa</i>)	Elimina blank finali
String (<i>numero</i> , <i>carattere</i>)	Stringa di <i>numero</i> caratteri indicati
Trim (<i>stringa</i>)	Elimina blank iniziali e finali
UCase (<i>stringa</i>)	Converte in maiuscolo

Funzioni di Formattazione

La funzione generica **Format** non esiste; al suo posto esistono FormatCurrency, FormatDateTime, FormatNumber, FormatPercent. Tutte queste funzioni per il formato fanno riferimento alle impostazioni internazionali del Sistema quindi potrebbero dare risultati diversi a seconda che lo script sia eseguito sul client o sul server.

Il valore restituito è una stringa.

FormatDateTime(Data[,formato])

Se formato è omissso viene utilizzato VbGeneralDate (se solo Data formato data breve, se solo Ora formato ora estesa, se entrambe data breve + ora estesa); tenere presente che la data è la parte intera della variabile tipo DateTime e la parte decimale ne è il tempo in frazione di giorno.

0 – VbGeneralDate

1 – VbLongDate: data in formato esteso

2 – VbShortDate: data in formato breve

3 – VbLongTime: ora in formato esteso

4 – VbShortTime: ora in formato hh:mm 24 ore.

Le altre tre funzioni hanno tutti gli stessi parametri:

FormatXXXXXX(Espressione[,NumCifreDecimali[,CifraIniziale[,NumeriNegativiTraParentesi[,CifreRaggruppate]]]])

Se un parametro è omissso vengono utilizzate le impostazioni di sistema.

CifraIniziale specifica se mettere 0 prima del Decimal Point nei numeri frazionari e CifreRaggruppate specifica se raggruppare le cifre della parte intera con separatore migliaia; i valori possibili per i 3 ultimi parametri sono:

-1: True

0: False

-2: impostazioni del sistema.

Esempio: FormatPercent(.125, 2) da "12,50%"; FormatNumber(.17, 3, -1) da "0,170";

FormatNumber(1793.2, 2, -1, 0, -1) da "1.793,20"

se nelle impostazioni del sistema il separatore decimale è la virgola.

Funzioni di Conversione

CBool (<i>espressione</i>)	Boleana (0 od 1, False o True)
CByte (<i>espressione</i>)	Byte
CDate (<i>espressione data</i>)	Data
CDBl (<i>espressione</i>)	Virgola mobile Double precision
CInt (<i>espressione</i>)	Intero
CLng (<i>espressione</i>)	Intero lungo
CSng (<i>espressione</i>)	Virgola mobile Single precision
CStr (<i>espressione</i>)	Stringa
Hex (<i>espressione</i>)	Esadecimale
Oct (<i>espressione</i>)	Ottale
Str (<i>espressione</i>)	Stringa – funzione generica
Val (<i>espressione</i>)	Numero – funzione generica

Poiché le variabili sono sempre di tipo Variant le funzioni di conversione sono utili per ottenere una variabile del sottotipo richiesto.

Attenzione agli argomenti delle funzioni: se sono stringhe devono rispettare le impostazioni internazionali del Computer su cui viene eseguito lo script.

Funzioni Logiche

IsArray (<i>variabile</i>)
IsDate (<i>espressione</i>)
IsEmpty (<i>espressione</i>)
IsNull (<i>espressione</i>)
IsNumeric (<i>espressione</i>)
IsObject (<i>espressione</i>)
StrComp (<i>stringa1</i> , <i>stringa2</i> [, <i>compare</i>])
VarType (<i>variabile</i>)

Le Funzioni **IsXXXX** restituiscono True|False se l'argomento è del tipo indicato o meno.

La Funzione **StrComp** effettua il confronto restituendo -1 se *stringa1* < *stringa2*, 0 se uguali, +1 se *stringa1* > *stringa2*, Null se una delle 2 è Null; il tipo di confronto è determinato dall'argomento *compare*:

Costante	Valore	Descrizione
vbUseCompareOption	-1	Esegue un confronto utilizzando l'impostazione dell'istruzione Option Compare .
vbBinaryCompare	0	Esegue un confronto binario.
vbTextCompare	1	Esegue un confronto di testo.
vbDatabaseCompare	2	Solo per MS Access; esegue un confronto in base alle informazioni del database.

La Funzione **VarType** restituisce i seguenti valori:

Costante	Valore	Descrizione
vbEmpty	0	Empty (non inizializzata)
vbNull	1	Null (dati non validi)
vbInteger	2	Intero
vbLong	3	Intero lungo
vbSingle	4	Numero a virgola mobile a precisione singola
vbDouble	5	Numero a virgola mobile a precisione doppia
vbCurrency	6	Valore di valuta
vbDate	7	Valore di data
vbString	8	Stringa
vbObject	9	Oggetto
vbError	10	Valore di errore
vbBoolean	11	Valore booleano
vbVariant	12	Variant (utilizzato solo con le matrici di variabili Variant)
vbDataObject	13	Oggetto di accesso ai dati
vbDecimal	14	Valore decimale
vbByte	17	Valore byte
vbArray	8192	Array

Se trattasi di matrice viene restituita la somma 8192+tipo valori matrice; esempio:
matrice Variant = 8192+12, matrice di interi = 8192+2, matrice non inizializzata = 8192.

7 – Oggetti

Gli Oggetti hanno Metodi, Proprietà ed Eventi collegati.

Le Proprietà possono essere sia modificabili, sia di sola lettura; inoltre si può verificare il caso di proprietà inizialmente modificabili che divengono di sola lettura dopo l'esecuzione di un certo metodo sull'oggetto.

I metodi possono richiedere o meno parametri.

La sintassi è *oggetto.metodo[(parametri)]* o *oggetto.proprietà*.

L'unico oggetto implicito è l'oggetto **Err**.

I due metodi relativi sono Clear e Raise e le proprietà sono Description, Number, Source; esistono inoltre le proprietà HelpContext e HelpFile se si desidera gestire il tasto F1 nel Dialog Box dei messaggi di errore.

Esistono poi gli oggetti dell'ambiente in cui si opera, quindi quelli del **Document Object Model** se si opera sul Client, quelli propri del **Server** se si opera su quest'ultimo.

In entrambi gli ambienti è possibile inoltre utilizzare oggetti ActiveX creati da noi stessi o da terze parti.