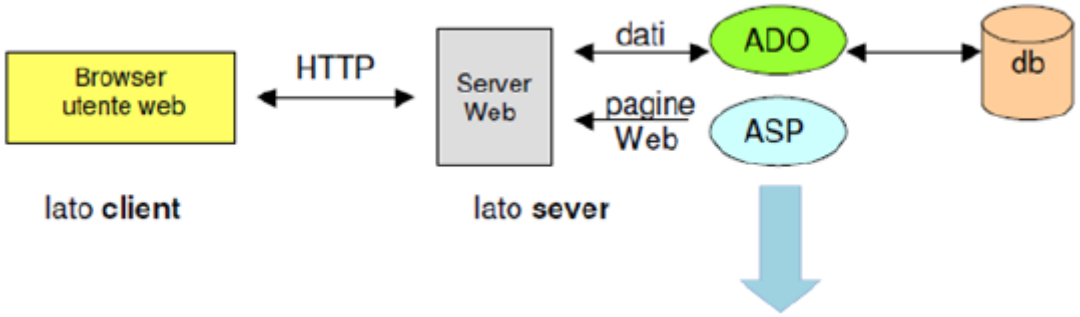
Sviluppo di applicazioni e servizi Web: istruzioni fondamentali VB

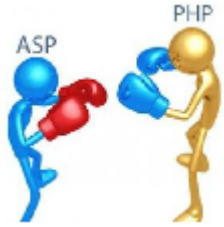
Gli sviluppatori possono realizzare applicazioni Web e servizi Web (Web Service) codificando pagine **ASP**[1] (*Active Server Pages*) o **ASPX** cioè usando quell'**insieme di** tecnologie di sviluppo di software per il web, commercializzate da Microsoft.

Pagine lato server *attive* (o dinamiche) perché integrano ***codice eseguibile*** a ***linguaggi di marcatura*** tipici nel realizzare pagine statiche (*form* di inserimento dati), ***oggetti*** per interagire con utenti (*client: il browser*) o con DB remoti (ADO cioè *ActiveX Data Objects*).

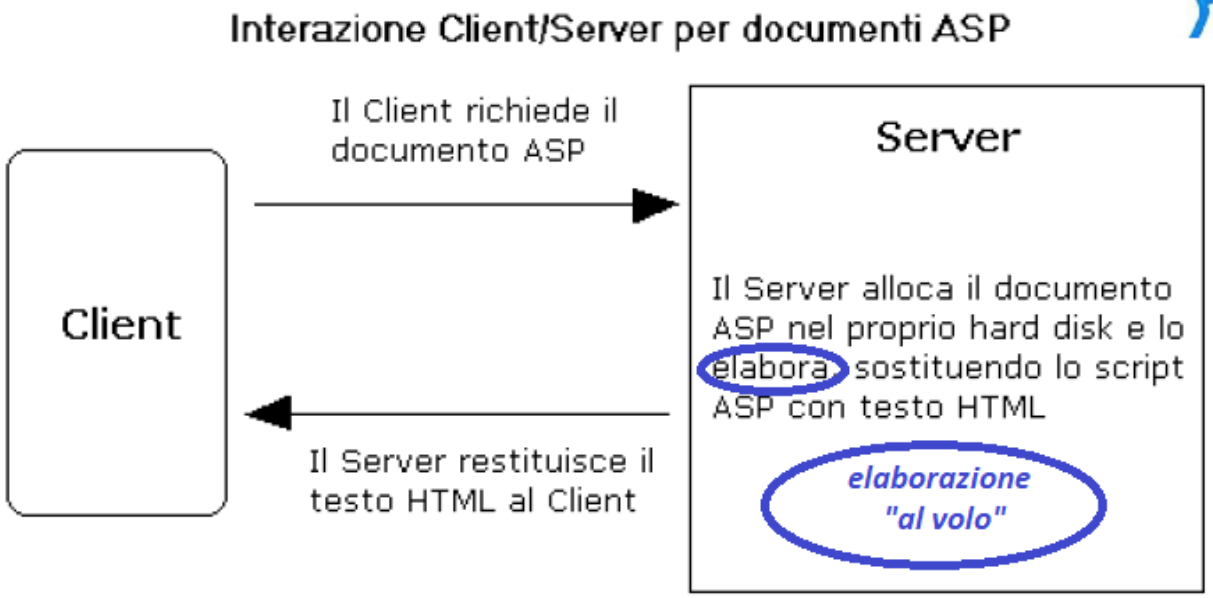


La filosofia è semplificare la migrazione degli sviluppatori dalle applicazioni Windows alle applicazioni web mettendoli in grado di generare pagine composte da tanti controlli *widget*, del tutto simili a quelli usati dall'interfaccia utente di Windows.

In realtà come ***codice eseguibile*** può essere usato uno qualsiasi dei linguaggi di alto livello supportati dal **framework** (libreria di classi) .NET, come, ad esempio, **Visual Basic** (linguaggio *event driven* la cui sintassi deriva dal BASIC, proprietario Microsoft, basato su oggetti e compilato), C# (linguaggio soggetto a una specifica di standardizzazione), J# (derivato da Java), ma anche, molti altri linguaggi open source (e non), come Perl, Python o JavaScript.



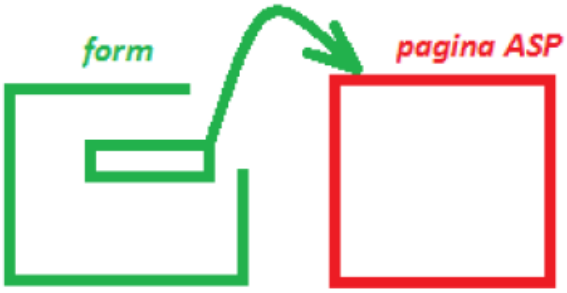
Le applicazioni ASP.NET sono significativamente più veloci e performanti rispetto a quelle realizzate utilizzando altre tecnologie di scripting, in quanto l'intero codice del sito web è pre-compilato in pochi file *dll* (spesso in un unico file) gestiti da un server Web.



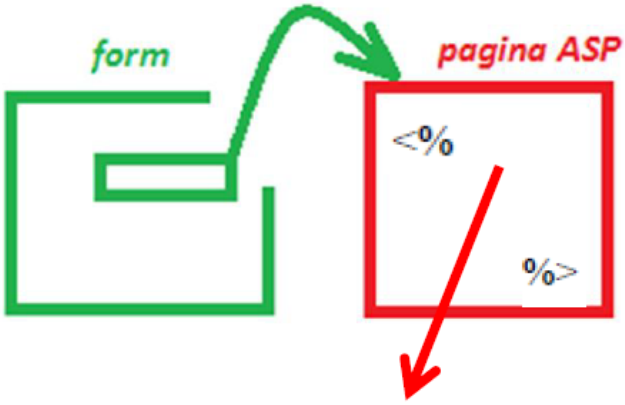
1 ASP vs ASPX. Utile tutorial navigando nel sito **w3schools.com** (*inglese*)

Esaminandone "l'anatomia", possiamo dividere una pagina ASP in tre parti:

1) Testo
2) Marcatori HTML e proprietà CSS
3) ***Oggetti*** per *ricevere/interpretare* dati e ***codice eseguibile*** per *maneggiare* i dati



pagina statica



pagina dinamica

codice eseguibile

pagina statica ***invioDati.htm***	*pagina dinamica* ***leggi.asp*** *che genera al volo codice HTML*

<html>
<head><title>Censimento</title></head>
<body>
 Censimento
 Benvenuti. vi preghiamo di compilare la scheda seguente
 per inoltrare la scheda usare il pulsante inoltra la scheda
<hr>
 	
<form method="get" action ="***leggi.asp***">
 nome: <input type = "text" name = "nome">
 sesso:
<input type = "radio" name = "sesso" value = "m">maschile
<input type = "radio" name = "sesso" value = "f">femminile
<input type = "radio" name = "sesso" value = "n">nullo
 <input type = "submit" value = "inoltra la scheda ">
 <input type = "reset" value = "annulla la scheda ">
</form>
 	
<hr>
</body>
</html>
	<html>
	<head><title>Active Server Page</title></head>
	<body>
	 Grazie per aver inviato i dati richiesti
	
	<%
	' lettura da form
	' che acquisisce il nome ed il sesso
	rem per default l'esecuzione é sul server
	rem ed il linguaggio dello script é *VisualBasic*
	
	if request.querystring ("**sesso**") = "f" **then**
	response.write (" benvenuta ")
	
	elseif request.querystring ("**sesso**") = "m" **then**
	response.write (" benvenuto ")
	
	else
	
	response.write (" salve ")
	end if
	response.write (request.querystring ("**nome**"))
	
	' eventuali ulteriori elaborazioni
	%>
	
	</body>
	</html>

Esempio di pagina **HTML** con **form** archiviata su server con stesso **percorso** della **pagina *ASP***

Esempio di pagina che **elabora** con uso di ***If*** **nidificati**

Istruzioni fondamentali - VBScript

Cominciamo con il vedere come sia possibile in ***ASP***, implementare il costrutto ***alternativa*** con istruzioni **condizionali** o compiere **operazioni cicliche** utilizzando le istruzioni di iterazione messe a disposizione (*VBScript*, abbreviazione di Microsoft's Visual Basic Scripting Edition, è un sottoinsieme di *Visual Basic*). Esse sono di tre tipi: DO ….. UNTIL (o WHILE) LOOP, WHILE….WEND e FOR….NEXT. Vediamo un esempio per ognuna:

Esempio con "Do while …loop"	**Commenti**

<html><head>
<title>Es_asp1</title></head>
<body bgcolor="#FFFFFF">
<p>Do Loop con while
 	
<%
counter = 1
 	
thismonth = ***month***(***now***())
 	
Do while counter < thismonth + 1
  response.write "month number " **&** counter **&** " "
 	
  response.write "_____" **&** "
 	
"
 	
  If counter >13 ***then***
  exit do
  end if
  counter = counter+1
Loop
%>
<hr></body></html>
	month(now()) restituisce il numero del mese corrente
	
	Do while esegue mentre la condizione é *true*
	
	In una stringa occorre concatenare con **&**
	
	La ***if … then…end if***
	
	L'istruzione ***exit*** forza l'uscita da un ciclo

In questo esempio è stata introdotta anche l'istruzione ***If…Then…Else…End If***

Esempio con "Do until …loop"	**Commenti**

<html><head><title>Es_asp2</title></head>
<body bgcolor="#FFFFFF">
<p>Do Loop con until
 	
<%
counter = 1
thismonth = ***month***(***now***())
Do until counter = thismonth + 1
  response.write "month number " **&** counter **&** " "
 	
  response.write "_____" **&** "
 	
"
 	
  If counter >13 ***then***
  exit do
  end if
  counter = counter+1
Loop
%>
<hr></body></html>
	Do until esegue finché la condizione diventa *true*

È anche utile mostrare un esempio dell'istruzione ***Select Case***, che permette di sveltire la scrittura del codice, nel caso in cui ci siano diverse istruzioni if da svolgere:

Esempio con "Select case"	**Commenti**
<html> <head> <TITLE>Es_asp5</TITLE> </head> <body bgcolor="#FFFFFF"> Inserisci un numero da 0 a 4 <form> Inserisci il tuo grado di stipendio (0-4): <input type="text" name="grado"> <input type="submit" name="Invia"> </form> <% dim GradoSalario GradoSalario=request.querystring("grado") ***Select Case*** GradoSalario ***case*** 0,1 response.write("Poveraccio...") ***case*** 2,3 response.write("Buongiorno Signore !") ***case*** 4 response.write("Eccellenza come va ?") ***End Select*** %> </body> </html>	

Continuiamo, presentando ora un esempio sull'istruzione, ***For…Next,*** diviso in più parti e in cui troviamo tutti i possibili utilizzi dell'istruzione:

Esempio con "For…Next"	**Commenti**
<HTML> <HEAD><TITLE>Es_asp3</TITLE></HEAD> <BODY bgcolor="#FFFFFF"> Esempi di ***for..next*** <% ***for*** counter = 1 ***to*** 5 response.write "ora siamo in loop" & " " ***next*** %> <% ***for*** counter = 1 ***to*** 5 response.write "ora siamo al ciclo "_ & counter & " " ***next*** %>	In questa prima parte, l'uso di ***for next***, è quello tradizionale **"_ "** per continuare l'istruzione, andando a capo

 Esempio di ***for..next*** con ***step*** <% ***for*** counter = 0 ***to*** 25 ***step*** 5 response.write " ora siamo al ciclo "_ & counter & " " ***next*** %> Esempio di ***for..next*** con conto alla rovescia <% ***for*** counter = 50 ***to*** 25 ***step*** –5 response.write " ora siamo al ciclo "_ & counter & " " ***next*** %> </BODY></HMTL>	In questo caso invece l'incremento avviene specificando il passo attraverso l'istruzione ***step*** L'istruzione ***step*** può essere usata anche per un "conto alla rovescia"

Concludiamo questa prima carrellata di esempi, con quello che riguarda il ciclo WHILE…WEND e l'esempio di come sia possibile effettuare un "***ciclo infinito***" con l'istruzione DO…LOOP:

Esempio con "While….Wend"	**Commenti**
<HTML><HEAD> <TITLE>Es_asp4</TITLE></HEAD> <body bgcolor="#FFFFFF"> <% cont=0 WHILE cont < 5 Response.write "Ciao a tutti!" Cont=cont+1 WEND %> </BODY></HTML>	esegue mentre la condizione é *true*

Esempio con "Do…Loop"	**Commenti**
<HTML><HEAD> <TITLE>Ciclo infinito</TITLE></HEAD> <body bgcolor="#FFFFFF"> <%DO%> **' not ready yet!** <%LOOP%> </BODY></HTML>	In questo esempio non c'è nessun modo per uscire dal ciclo se non cambiare la pagina! (***Attenzion***e: il processo rimane attivo sul server a meno che l'amministratore non lo uccida!)

É interessante anche notare come in questo caso, sia usato ASP, in ***piena fusione*** con l'HTML.

Infatti le *istruzioni* in ASP (cioè il codice eseguibile integrato in pagina ASP) sono tutte racchiuse all'interno dei tag **<% %>**: in questo modo è possibile usare i tag HTML e le *istruzioni*.

Diamo infine, un'occhiata a come gestire gli **array**: esiste la possibilità di gestire ***collezioni di dati***, statiche o provenienti da form.

La sintassi per ***dichiarare*** un ***array monodimensionale*** è la seguente:

Dim *nome* (*bound*)

nome	Obbligatorio. Nome della matrice.
bound	*Opzionale*. Limite relativo alla dimensione della matrice che si definisce: numero elementi - 1

Esempi:

Metodo 1: Usando ***Dim nome( )***
Dim arr1**( )** **'senza dimensione**

Metodo 2 : Esplicitando le dimensioni ***Dim nome(bound)***
Dim arr2 (5) **'Dichiarazione di array di 6 elementi: indice da 0 a 5**

Metodo 3 : Passando il valore degli elementi come parametri alla ***funzione Array***
Dim arr3
arr3 = Array("Apple","Orange","Grapes")

Metodo 4: Inizializzando il valore degli elementi al momento della dichiarazione
Dim arr4 = {4, 5, 6}

Esempio ***Array monodimensionale***

```
<!DOCTYPE html>
<html>
  <body>
    <script language = "vbscript" type = "text/vbscript">
      Dim arr (5)          ' 6 elementi
      arr(0) = "1"              'Number as String
      arr(1) = "VBScript"       'String
      arr(2) = 100              'Number
      arr(3) = 2.45             'Decimal Number
      arr(4) = #10/07/2013#     'Date
      arr(5) = #12.45 PM#       'Time

      document.write("Value stored in Array index 0 : " & arr(0) & "<br />")
      document.write("Value stored in Array index 1 : " & arr(1) & "<br />")
      document.write("Value stored in Array index 2 : " & arr(2) & "<br />")
      document.write("Value stored in Array index 3 : " & arr(3) & "<br />")
      document.write("Value stored in Array index 4 : " & arr(4) & "<br />")
      document.write("Value stored in Array index 5 : " & arr(5) & "<br />")
    </script>
  </body>
</html>
```

La sintassi per ***ridimensionare*** un ***array monodimensionale*** è la seguente:

ReDim *nome (bound)*

nome	Obbligatorio. Nome della matrice.
bound	Obbligatorio. Limrte relativo alla dimensione della matrice ridefinita.

Limite di matrice: *boundlist* è in grado di specificare il limite inferiore e superiore della dimensione. Il limite inferiore è sempre pari a 0 (zero). *Il limite superiore rappresenta il* ***valore di indice più alto possibile per la dimensione****, non la lunghezza della dimensione.*

Ulteriore ***modificatore*** *opzionale*

ReDim Preserve *nome (bound)*

preserve	Modificatore utilizzato per conservare i dati nella matrice esistente quando si modifica soltanto la grandezza dell'ultima dimensione

Esempio: **ReDim *Preserve*** *numbers (*15*)* sostituisce l'array di nome *numbers* preservandone i valori dei **16** elementi

Per array monodimensionali *l'istruzione*

*NomeArray.Resize(*A, n*)* con n si intende il numero di elementi

equivale, usando ***ReDim***, a:

ReDim A(n – 1) con n-1 si intende l'indice massimo

Esempio ***Ridimensionamento di array***

```
<!DOCTYPE html>
<html>
  <body>
    <script language = "vbscript" type = "text/vbscript">
      Dim a( )      ' nessun elemento
      i = 0
      ReDim a(5)        ' aumento a 6 elementi
      a(0) = "XYZ"
      a(1) = 41.25
      a(2) = 22

      ReDim Preserve  a(7)  ' aumento a 8 elementi  mantenendo  i valori degli elementi con indici 0-2
      For i = 3 to 7
            a(i) = i     ' inizializzo con il valore dell'indice gli elementi a(3) - a(7)
      Next

      ' per verificare,  visualizzo in output
      For i = 0 to Ubound (a)                    ' funzione cherecupera l'indice massimo
        Msgbox a(i)
      Next
    </script>
  </body>
</html>
```

Array a più dimensioni

La sintassi per ***dichiarare*****:** ***Dim*** *nome (bound1, bound2, ...)*

Ad esempio: ***Dim*** *Array* ***(****dim1, dim2****)***

In questo caso abbiamo definito un array *bidimensionale* con *dim1+1* **righe** *e dim2+1* **colonne**. Per accedere ai suoi elementi basta inserire i valori degli indici, ricordando che essi partono dallo zero fino a *dim1* o *dim2.*

Esempio ***Array a più dimensioni***

```
<!DOCTYPE html>
<html>
  <body>
    <script language = "vbscript" type = "text/vbscript">
      Dim arr (2, 3)      ' 3 righe e 4 colonne
      arr(0,0) = "Apple"
      arr(0,1) = "Orange"
      arr(0,2) = "Grapes"
      arr(0,3) = "pineapple"

      arr(1,0) = "cucumber"
      arr(1,1) = "beans"
      arr(1,2) = "carrot"
      arr(1,3) = "tomato"

      arr(2,0) = "potato"
      arr(2,1) = "sandwitch"
      arr(2,2) = "coffee"
      arr(2,3) = "nuts"

      document.write("Value in Array index 0,1 : " &  arr(0,1) & "<br />")
      document.write("Value in Array index 2,2 : " &  arr(2,2) & "<br />")

    </script>
  </body>
</html>
```

La sintassi per ***ridimensionare:***

ReDim [Preserve] *nome* (boundlist*1*, boundlist*2* [, ...])

Limiti di matrice. Ogni boundlist*n* è in grado di specificare i limiti inferiori e superiori della dimensione*n*. Il limite inferiore è sempre pari a 0 (zero). Il limite superiore rappresenta il ***valore di indice più alto possibile per la dimensione***, non la lunghezza della dimensione.

Nb: uso dei simboli [e] per evidenziare l'opzionalità

Più variabili. È possibile ridimensionare più variabili di matrice nella stessa istruzione di dichiarazione e specificare le parti di *boundlist* e di *name* per ogni variabile. Le variabili sono separate da una virgola

ReDim [Preserve] *name1* (boundlist*1*) [, *name2* (boundlist*2*) [, ...]]

Appendice

Inbuilt functions VBScript per elaborare array

Function	Description
LBound	A Function, which returns an integer that corresponds to the **smallest** subscript of the given arrays.
UBound	A Function, which returns an integer that corresponds to the **largest** subscript of the given arrays.
Split	A Function, which returns an array that contains a specified number of values. Splitted based on a Delimiter.
Join	A Function, which returns a String that contains a specified number of substrings in an array. This is an exact opposite function of Split Method.
Filter	A Function, which returns a zero based array that contains a subset of a string array based on a specific filter criteria.
IsArray	A Function, which returns a boolean value that indicates whether or not the input variable is an array.
Erase	A Function, which recovers the allocated memory for the array variables.

Utile tutorial nel sito w3schools.com (*inglese*)

- Tutte le funzioni *VBScript* nello sviluppo di applicazioni web con uso di tecnologia ASP

• Date/Time functions • Conversion functions • Format functions	• Math functions • Array functions	• String functions • Other functions

Operatori, tipi di dato e funzioni di conversione in VBScript

Un ruolo importante, per ogni linguaggio di programmazione, è rivestito dagli **operatori** e dai **tipi di dato**; VBScript mette anche a disposizione una serie di ***funzioni predefinite*** che servono ad attuare la conversione di un dato da un tipo all'altro. Queste funzioni prendono il nome di **funzioni di conversione**.

I **tipi di dato** che *VBScript* mette a disposizione sono i seguenti:

Tipo di dato	Descrizione
Boolean	Può assumere i valori True o False
Byte	Numerico intero compreso in un intervallo che va da 0 a 255
Currency	Utilizzato per esprimere valore monetario; va da -922.337.203.685.477,5808 a 922.337.203.685.477,5807
Date	Utile per trattare le date; compreso in un intervallo che va dal 1 Gennaio 100 al 31 Dicembre 9999
Double	Numerico in virgola mobile a precisione doppia, compreso in un intervallo che va da -1,79769313486232E308 a 1,79769313486232E308
Integer	Numerico intero compreso in un intervallo che va da -32.768 a 32.767
Long	Numerico intero compreso in un intervallo che va da -2.147.483.648 a 2.147.483.647
Object	Può contenere un oggetto, ad esempio di tipo ActiveX
Single	Numerico in virgola mobile a precisione singola, compreso in un intervallo che va da -3,402823E38 a 3,402823E38
String	Adatto per un valore di tipo stringa testuale di lunghezza variabile, fino ad un massimo di circa 2 miliardi

Gli ***operatori*** che VBScript mette a disposizione:

Operatori aritmetici

\+ Esegue una somma
\- Esegue una sottrazione
* Esegue una moltiplicazione
/ Esegue una divisione
\ Esegue una divisione intera → restituisce un numero intero che indica quante volte è contenuto un numero in un altro
^ Elevamento a potenza
Mod Modulo di una divisione (cioè il resto intero delladivisione fra interi)

Operatori di assegnamento

= Assegnazione di un valore (*lo stesso simbolo è usato anche per l'operatore di uguaglianza*)

Operatori di confronto

= uguale
<> Diverso
> Minore
> Maggiore
<= Minore o uguale
>= Maggiore o uguale

Operatori logici

Not Negazione
And Congiunzione Logica
Or Disgiunzione Logica
Xor Or esclusivo
Eqv Equivalenza logica: entrambi valore falso o vero
Imp Implicazione logica: uno è falso l'altro è vero (A Imp B **TRUE quando...** A è False o B è True **FALSE quando...** A è True o B è False)

Operazioni di Assegnazione composta

Consentono di compattare la scrittura del codice scrivendo formule del tipo:

Variabile = Variabile **operatore** espressione

Nel seguente modo: Variabile **operatore=** espressione

Operatore	Esempio	Corrisponde a
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
\=	a\=b	a=a\b
^=	a^=b	a=a^b
&=	a&=b	a=a&b (concatenazione di stringhe)

La negazione[2] può utilizzare lo stesso simbolo della sottrazione - se serve per convertire un numero positivo in negativo e viceversa. Equivale insomma a moltiplicare per -1 il valore in questione.

In funzione di questi tipi di dato esistono una serie di **funzioni di conversione** per ritipizzare una variabile.

Di seguito un elenco:

Funzione	Descrizione
Abs	Restituisce il valore assoluto di un numero
CByte	Converte un tipo in un tipo Byte
CDate	Converte un tipo in un tipo Date
CDbl	Converte un tipo in un tipo Double
CInt	Converte un tipo in un tipo Integer
CLng	Converte un tipo in un tipo Long
CSng	Converte un tipo in un tipo Single
CStr	Converte un tipo in un tipo String
Fix	Restituisce la parte fissa di un numero
Hex	Restituisce, in formato String, il valore esadecimale di un numero
Int	Restituisce la parte intera di un numero
Oct	Restituisce, in formato String, il valore ottale di un numero
Round	Arrotonda un numero ai decimali specificati
Sgn	Restituisce un intero che rappresenta il segno di un numero

Per approfondimento: altra dispensa online

2 Tratto da pdf scaricabile “Introduzione alla programmazione ASP” del prof. G.Tombolini