ASP
(Active Server Pages)

Ing. Marius Spinu

2000

---

1. Form ed interazioni con il server

1.1. Introduzione ad ASP

Le Active Server Pages (**ASP**) sono inserite nei file con estensione .asp.
Un file .asp è un file testuale che contiene:

- Testo
- Tag HTML
- Comandi di Script.

Un comando di script istruisce il computer a fare qualcosa, come assegnare un valore ad una variabile. E' alquanto facile creare un file .asp: Basta rinominare un qualsiasi file HTML, sostituendo i file con le estensione esistenti .htm o .html con l'estensione .asp.

Per creare uno script .asp disponibile agli utenti del Web, bisogna salvare il file in una directory appartenente al Web server ed essere sicuri di avere il permesso d esecuzione degli script. Quando il file verrà chiamato dal browser il processo ASP restituirà codice HTML.

Uno **Script** è composto da una serie di comandi di Script.

L'esecuzione di uno script invia una serie di comandi ad un motore di scripting (scripting engine), che lo interpreta. Gli Script sono scritti in linguaggi aventi precise regole; tra queste: per poter usare un linguaggio di scripting, il server deve disporre di uno scripting engine che conosca il linguaggio. ASP contiene gli scripting engines per i linguaggi di scripting VBScript and JScript. Il primo linguaggio di scripting che ASP assume, se non altrimenti stabilito, è VBScript.

ASP in realtà non è un linguaggio di scripting ma offre un environment (ambiente) che processa gli script che sono incorporati nelle pagine HTML.

Delimitatori ASP

I Tag HTML sono differenziati dal testo dai delimitatori. Un delimitatore è un carattere o una sequenza di caratteri che marcano l'inizio e la fine di una unità.
Nel caso dell'HTML, questi delimitatori sono il simbolo minore (<) e maggiore (>).

In modo del tutto simile, i comandi e le espressioni di output degli script ASP sono differenziati sia dal testo che dai Tag HTML dai delimitatori. ASP usa i delimitatori ***<% e %>*** per includere i comandi di script.
Ad esempio, il comando ***<% nMioNumero = 5 %>*** assegna il valore numerico 5 alla variabile nMioNumero.

ASP usa i delimitatori ***<%= e %>*** per racchiudere le espressioni di output.
Ad esempio, l'espressione di output ***<%= nMioNumero %>*** invia il valore 5 (valore corrente della variable) al browser.

1.2. Lettura da form

Esempio file html con form:

```
<html>
<head>
<TITLE>form</TITLE>
</head>
<body bgcolor="#FFFFFF">
<form action="respond1.asp" method=get>
Your First Name<INPUT NAME="FirstName" MaxLength=20><p>
Your Last Name<INPUT NAME="LastName" MaxLength=20><p>
<INPUT TYPE=submit><p>
<INPUT TYPE=reset>
</form>
</body>
</html>
```

Si noti la presenza di due form di tipo testo chiamate **FirstName** e **LastName**.
Ed il generatore di risposta in ASP:

```
<html>
<head>
<TITLE>respond1.asp</TITLE>
</head>
<body>
<%
fname=request.querystring("Firstname")
lname=request.querystring("Lastname")
%>


<%If (fname="" or lname="") then%>
Non hai inserito i tuoi dati. <p>
<% Else %>
Ciao <%=fname%> <%=lname%><p>
<%end if%>

</body>
</html>
```

Il file ASP non fa altro che leggere i dati dalle form e assegnarle a due variabili “fname” e “lname”.
Successivamente si verifica se le variabili sono vuote o no e tramite l’utilizzo di un costrutto
IF … ELSE … END IF si decide come proseguire.

Qualcosa di simile si può fare anche utilizzando il costrutto CASE:

```
<html><head>
<TITLE>esempio1.asp</TITLE>
</head>
<body>
<%fname=request.querystring("Firstname")
lname=request.querystring("Lastname")
```

```
response.write "Nice to Hire You " & fname & " " & lname & "<p>"
Select Case lcase(fname)
      case "pippo","pluto","",
            response.write("Non hai inserito il nome o il nome è fasullo")
      case "Maria","Mario","Marco",
            response.write("Mi piace il tuo nome")
      case "Nostradamus"
            response.write("Hai un nome un po' strano")
End Select%>
</body></html>
```

Si osservi in questo caso oltre all'utilizzo del CASE anche il modo diverso con qui si passa un dato al browser tramite la funzione response.write().

Ovviamente si possono leggere anche altri tipi di form come le checkbox:

```
<html>
<head>
<TITLE>FormCheckBox.html</TITLE>
</head>
<body>
<form action="leggiform.asp" method="post">
<p>CheckBox Form</p>
<p>Fai la tua scelta sul modo in qui puoi essere contattato:</p>
<input TYPE="checkbox" NAME="posta">Poste Italiane<br>
<input TYPE="checkbox" NAME="corierre">Lettera via corriere<br>
<input TYPE="checkbox" NAME=”email" CHECKED>per email<br>
<input TYPE="checkbox" NAME="fax">per Fax<br>
<input TYPE="checkbox" NAME="tel" CHECKED>telefono<br><br>
<input type="submit"><input type="reset">
</form>
</body>
</html>
```

Un possibile modo per analizzare il form precedente può essere:

```
<html>
<head>
<TITLE>formCheckBoxRespond.asp</TITLE>
</head>
<body>
<%
If request.form("posta")="on" then
        response.write "<br>Ti manderò una lettera con la posta
end if
If request.form("corriere")="on" then
        response.write "<br>Ti manderò una conferma con il corriere
end if
If request.form("email")="on" then
        response.write "<br> Ti manderò un messaggio email
end if
If request.form("fax")="on" then
        response.write "<br>Ti confermo per fax
end if
If request.form("tel")="on" then
        response.write "<br>Ti chiamerò al telefono
end if%>
</body>
</html>
```

Nel caso di un form di select (o equivalente per RadioBox):

```
<SELECT NAME="state">
<OPTION SELECTED VALUE="md">Maryland</OPTION>
<OPTION value="dc">District of Columbia</OPTION>
<OPTION value="va">Virginia</OPTION>
<OPTION value="whoknows">Somewhere Else!</OPTION>
</SELECT>
```

il file asp può essere (nel codice non è stata inserita la parte HTML):

```
<%mystate = request.form("state")
Select Case ucase(mystate)
case "DC"
        response.write "DC meeting on 15th"
case "MD"
        response.write "MD meeting on 20th"
case else
        response.write "No meetings for you to attend!"
End Select%>
```

***NOTA**: Si osservi l'utilizzo di lcase e ucase che permettono un confronto migliore (lcase trasforma il testo in lettere minuscole e ucase in lettere maiuscole).*

2. Cicli e selezioni

2.1. IF

```
If condizione=vera then
        response.write "E vero"
else
        response.write "E falso"
end if
```

dove ovviamente la parte else può mancare.
La condizione più complessa ha la forma (riferendosi all'esempio iniziale con due form che chiedono nome e cognome):

```
<%If fname="george" and lname="washington" then%>
      Ciao.<p>Tu sei il primo presidente americano!
<%elseIf fname="bill" and lname="clinton" then%>
      Ciao.<p>Tu sei il presidente americano!
<%elseIf fname="diego" and lname="maradona" then%>
      Ciao.<p>Sei stato un bravo calciatore
<%elseIf fname="John" or fname="lennon" then%>
      Ciao.<p>Sei stato uno dei Beatles!
<%else%>
      Ciao!<p> Non ti conosco.
<%end if%>
```

2.2. Ciclo FOR

For con passo 1:

```
<%for numero = 1 to 5
response.write "siamo in un loop" & "<br>"
next%>
```

For con passo 5:

```
<%for numero = 25 to 50 step 5
response.write "siamo nel loop numero " & numero & "<br>"
next%>
```

For con passo -5:

```
<%for numero = 50 to 25 step -5
        response.write "siamo nel loop numero " & numero & "<br>"
next%>
```

2.3. Ciclo WHILE

```
<%
numero = 1
questomese = month(now())
Do while numero < questomese + 1
response.write "<br>il mese numero" & numero & "<br>"
If numero >13 then
        exit do
```

```
end if
numero = numero+1
Loop
%>
```

2.4. SELECT CASE

```
<%
Select Case lcase(fname)
      case "pippo","pluto",""
            response.write("Non hai inserito il nome o il nome è fasullo")
      case "Maria","Mario","Marco",
            response.write("Mi piace il tuo nome")
      case "Nostradamus"
            response.write("Hai un nome un po' strano")
End Select
%>
```

2.5. FUNZIONI e SUBROUTINE

```
<%
Function ODBCLng(TheVal)
      If TheVal = "" Or IsNumeric(TheVal) = False Then
            ODBCLng = "0"
      Else
            ODBCLng = TheVal
      End If
End Function
%>

<%
Sub ShowFolderList(folderspec)
    Dim fs, f, f1, fc, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(folderspec)
 Set fc = f.SubFolders
    For Each f1 in fc
  s = s & f1.name
  s = s &  "<BR>"
 Next
    Response.Write s
End Sub%>
```

2.6. Qualche istruzione particolare.

```
<%
strDate = weekdayname( weekday(date) ) & " "
strDate = strDate & monthname( month(date), true ) & " "
strDate = strDate & day(date) & ", "
strDate = strDate & year(date)
%>
```

Con l'istruzione include il contenuto del file (header.asp) è eseguito nel posto dove è stata effettuata la chiamata:

```
<!--#include file="header.asp"-->
```

La parola chiave Virtual è usata per indicare un percorso che inizia con una directory virtuale.

```
<!--#INCLUDE VIRTUAL="/miadirectory/fileincluso.inc"-->
```

La parola chiave File è usata per indicare un percorso relativo. Un percorso relativo inizia con la directory che include il file.
Per esempio, se si ha un file nella directory MiaDirectory, ed il file FileMio.inc è nella directory MiaDirectory\FileInclusi, la seguente linea inserisce il file FileMio.inc nel file:

```
<!--#INCLUDE FILE="FileInclusi/FileMio.inc"-->
```

Notare che il percorso per includere il file, FileInclusi/FileMio.inc, è relativo al file da includere.
E' anche possibile usare il parametro FILE con la sintassi ../ per includere un file da una directory di livello superiore, se la chiave del Registry EnableParentPaths è settata ad 1.

Un file incluso può includere altri file. Un file .asp può includere lo stesso file più di una volta, l'importante è che l'istuzione <INCLUDE> non causi un ciclo infinito.

Si suppone di avere una stringa come (letta da una form e assegnata alla variabile strfull):
`informatica, dizionario, HTML, JAVA, JavaScript, VBasic, CSS`
e si vogliono trovare le parole separate da virgola per creare un indice di parole chiave.

```
<%
public namearray
namearray=split(strfull,", ")
max=ubound(namearray)
'response.write "namearray=" & namearray(0) & "<p>"
'response.write "max=" & max & "<p>"
for indice= 0 to max
        response.write namearray(indice) & "<p>"
next
response.write "<hr>"
%>


strAuthorFull = Filter(strSingleArray,"InsertAuthors")
strAuthor = mid (strAuthorFull(0),inizstrNome+14)
inizpuntovirg = instr(strAuthor,"|")
```

3. Connessione ad un database

Si suppone di avere un file html con una sola form dove si deve inserire la chiave di ricerca su un campo **Keyword** di una tabella **KeyTable** di un database Access chiamato **db1.mbd.** Si vuole elencare il contenuto del campo **Description** del database.

```
<html>
<head>
        <title>update</title>
</head>

<body>
<%
Key=request.querystring("chiave")
‘ addKey è il nome della form
‘ ed il testo inserito è assegnato alla variabile Key
response.write key
‘ connect to database
        set conntemp=server.createobject("adodb.connection")
        conntemp.open "driver={Microsoft Access Driver (*.mdb)};DBQ=" +
Server.MapPath("..\db1.mdb")+""
        strSQL = "select Description from KeyTable where Keyword='" & Key &"'"
        response.write strSQL
        set rstemp = conntemp.execute(strSQL)

        do while not rstemp.eof %>
                <%= rstemp(0)%>
<%               rstemp.movenext
        loop
        rstemp.close
        set rstemp=nothing
        conntemp.close
        set conntemp=nothing

%>

</body>
</html>
```

dopo aver assegnato il testo letto alla variabile Key lo si scrive sullo schermo, si apre la connessione al database db1.mdb che si trova nella directory superiore alla directory corrente (in fatti si è inserito ..\db1.mdb). Successivamente è stata creata una stringa strSQL che contiene il commando SQL da utilizzare per fare la query:

select `Description from KeyTable where Keyword='" & Key &"'"`

dopo aver eseguito la query si scrive sullo schermo il contenuto della variabile rstemp(0) finchè la variabile risulta vuota, cioè non ci sono più elementi che soddisfano il criterio di ricerca.
Nella parte conclusiva si chiude la query rstemp e la connessione conntemp.

Oppure se si vogliono leggere più campi dello stesso record (come ID, nome e anno di nascita) della tabella "authors" (si suppone che la connessione sia già aperta):

```
<%
sqltemp="select * from authors where AU_ID=" & whichid
set rstemp=conntemp.execute(sqltemp)
```

```
writerID=rstemp("AU_ID")
writername=rstemp("Author")
writeryearborn=rstemp("Year Born")
%>
```

Si possono utilizzare i risultati della query per costruire delle nuove form:

```
<html><head>
<TITLE>/learn/test/dblist.asp</TITLE>
</head><body bgcolor="#FFFFFF">
<%
' displays a database field as a listbox
set conntemp=server.createobject("adodb.connection")
conntemp.open "DSN=Student;uid=student;pwd=magic"
set rstemp=conntemp.execute("select nome from Elenco where AU_ID<100")
%>
<Form><Select>
<% ' Now lets grab all the data
do while not rstemp.eof %>
    <option> <%=RStemp(0)%> </option>
<%
rstemp.movenext
loop
rstemp.close
set rstemp=nothing
conntemp.close
set conntemp=nothing
%>
</Select>
</form></body></html>
```

Ovviamente si possono eseguire anche scritture nel database tramite l'opportuna chiamata SQL:

```
<%
auname=request.querystring("name")
' fix embedded apostrophes
auname=Replace(auname, "'", "''")
auyear=request.querystring("year")
auID=request.querystring("ID")
Set Conn = Server.CreateObject("ADODB.Connection")
conn.open "DSN=Student;uid=student;pwd=magic"
'          DSN       user     password
SQLstmt = "UPDATE authors "
SQLStmt = SQLstmt & "SET Author='" & auname & "',"
SQLstmt = SQLstmt & "year born=" & auyear
SQLStmt = SQLStmt & " WHERE AU_ID=" & auid
Set RS = Conn.Execute(SQLStmt)
%>
```

In questo caso si modificano i campi del record che soddisfa la condizione impostata. Si osservi l'utilizzo della funzione Replace() che sostituisce gli apostrofi nella stringa da inserire.

```
<%
set conntemp=server.createobject("adodb.connection")
      conntemp.open "driver={Microsoft Access Driver (*.mdb)};DBQ=" +
Server.MapPath("..\db1.mdb")+""

      SQLStmt = "INSERT INTO authors (AU_ID,author,year_born) "
```

```
      SQLStmt = SQLStmt & "VALUES (" & auid
      SQLStmt = SQLStmt & ",'" & auname & "'"
      SQLStmt = SQLStmt & "," & int(auyear) & ")"
      Set RS = Conn.Execute(SQLStmt)

        conntemp.close
        set conntemp=nothing

%>
```

Eventualmente si può inserire un codice che controlla gli errori:

```
SQLStmt = "INSERT INTO authors (AU_ID,author,year_born) "
SQLStmt = SQLStmt & "VALUES (" & auid
SQLStmt = SQLStmt & ",'" & auname & "'"
SQLStmt = SQLStmt & "," & int(auyear) & ")"
Set RS = Conn.Execute(SQLStmt)
If err.number>0 then
   response.write "VBScript Errors Occured:" & "<P>"
   response.write "Error Number=" & err.number & "<P>"
   response.write "Error Descr.=" & err.description & "<P>"
   response.write "Help Context=" & err.helpcontext & "<P>"
   response.write "Help Path=" & err.helppath & "<P>"
   response.write "Native Error=" & err.nativeerror & "<P>"
   response.write "Source=" & err.source & "<P>"
   response.write "SQLState=" & err.sqlstate & "<P>"
else
   response.write "No VBScript Errors Occured" & "<P>"
end if
IF conn.errors.count> 0 then
   response.write "Database Errors Occured" & "<P>"
for counter= 0 to conn.errors.count
   response.write "Error #" & conn.errors(counter).number & "<P>"
    response.write "Error desc. -> " & conn.errors(counter).description & "<P>"
next
else
   response.write "No Database Errors Occured!" & "<P>"
end if
```

4. GLOBAL.ASA file

Il file Global.asa è un file opzionale nel quale si possono specificare script e dichiarare oggetti che hanno valore per la sessione o per l'applicazione corrente. Non è mai visibile all'utente, si deve chiamare global.asa e deve essere posizionato nella stessa directory dell'applicazione. Non si possono utilizzare più file global.asa per la stessa applicazione.

```
<script language=vbscript runat=server>
SUB Application_OnStart
END SUB

SUB Application_OnEnd
END SUB

SUB Session_OnStart
END SUB

SUB Session_OnEnd
END SUB
</script>
```

Appendice: Linguaggio Vbasic (parole chiave)

Funzioni

Abs	GetObject	Rnd
Array	Hex	Round
Asc	Hour	RTrim
Atn	InputBox	ScriptEngine
CBool	InStr	ScriptEngineBuildVersion
CByte	InStrRev	ScriptEngineMajorVersion
CCur	Int	ScriptEngineMinorVersion
CDate	IsArray	Second
CDbl	IsDate	Sgn
Chr	IsEmpty	Sin
CInt	IsNull	Space
CLng	IsNumeric	Split
Cos	IsObject	Sqr
CreateObject	Join	StrComp
CSng	LBound	StrReverse
CStr	LCase	String
Date	Left	Tan
DateAddFunction	Len	Time
DateDiff	LoadPicture	TimeSerial
DatePart	Log	TimeValue
DateSerial	LTrim	Trim
DateValue	Mid	TypeName
Day	Minute	UBound
Exp	Month	UCase
Filter	MonthName	VarType
Fix	MsgBox	Weekday
FormatCurrency	Now	WeekdayName
FormatDateTime	Oct	Year
FormatNumber	Replace	
FormatPercent	Right	

Metodi

Add	OpenTextFile	Skip
Clear	Raise	SkipLine
Close	Read	Write
CreateTextFile	ReadAll	WriteBlankLines
Exists	ReadLine	WriteLine
Items	Remove	
Keys	RemoveAll	

Constanti

Color	Miscellaneous
Comparison	MsgBox
Date/Time	String
Date Format	Tristate
File Input/Output	VarType

Operatori

+, -, /, *, ^, \ (integer division), AND, NOT, OR, XOR, Mod,