Corso di ASP

Lezione 1

Superare i limiti dell'html per creare dei siti sempre più rispondenti alle esigenze dei visitatori è stato una delle mete a cui i programmatori di linguaggi di scripting hanno puntato nel corso della storia del web. Dalle prime pagine statiche, manifesto di un sito, si è progressivamente arrivati non solo all'esplosione del multimediale, ma, soprattutto, al diffondersi di pagine interattive, in grado non solo di affascinare, ma di fornire un utile strumento a chi le volesse usare.
Di linguaggi scripting ne esistono parecchi, alcuni più simili a veri e propri linguaggi di programmazione, altri più facili. Il perl, ad esempio, è il tipico caso di linguaggio complesso, ma molto efficace, utilizzato per la creazione di cgi (common gateway interface) a livello professionale e poco diffuso fra gli utenti medi, a causa della difficoltà di apprendimento e di gestione della sua sintassi; inoltre, essendo nato in ambiente Unix ha trovato non poche difficoltà ad affermarsi al di fuori di una pur vasta cerchia di provider e professionisti del settore che utilizzano questo sistema operativo. Un'altra difficoltà notevole per l'utente medio è da sempre rappresentata dalla generale impossibilità di eseguire i cgi al di fuori della directory cgi-bin del web server del quale ci si serve, directory il cui accesso è limitato al web master. Solo in rari casi l'amministratore di sistema consente l'esecuzione di script perl (che hanno la possibilità di eseguire operazioni di lettura e scrittura su disco) da una qualsiasi directory. Se ciò ha una giustificazione a livello di sicurezza di un sito web, certo non ha invogliato gli utenti ad utilizzare questo linguaggio che richiede una "interazione" con provider e amministratori a volte troppo indaffarati per dare ascolto alle insolite richieste di un cliente.

Superare la staticità delle pagine web, mantenendo al contempo una semplicità di programmazione che consenta a tutti di intervenire senza prima dovere leggere voluminosi manuali è ora possibile grazie ai nuovi linguaggi di scripting. Fra tutti si distingue sicuramente l'asp (active server pages) per la rapidità e flessibilità di utilizzo che lo caratterizzano, che però sono controbilanciate da uno svantaggio non indifferente; l'utilizzo di questo linguaggio è confinato ai server Microsoft, come ad esempio a IIS, e non funziona quindi con tutti gli altri server che popolano il web. La sempre maggiore diffusione dei server Windows contribuisce però a rendere meno limitante questo ostacolo e, tutto sommato, non è difficile vedere diversi provider abbandonare il mondo Unix per le nuove possibilità offerte da Windows NT.

Grazie all'utilizzo delle pagine asp, l'utente può quindi creare dei documenti che possono fornire informazioni, rispondendo in modo diverso alle differenti richieste dei navigatori. Ma quali sono, in breve, i vantaggi nell'utilizzo di questo linguaggio di scripting?

1) Le pagine asp sono completamente integrate con i file html.
2) Sono facili da creare e non necessitano di compilazione.
3) Sono orientate agli oggetti e usano componenti server ActiveX.

Visti i vantaggi, e viste anche le limitazioni cui abbiamo accennato in precedenza, riassumiamo le tecnologie coinvolte nello sviluppo e funzionamento delle active server pages:

1) Windows NT
2) Protocollo tcp/ip
3) Un web server che supporti Active Server, come IIS
4) In via facoltativa, odbc (Open DataBase Connectivity) e un server database.

Esaminando più da vicino l'"anatomia" di questo genere di pagine possiamo constatare che esse sono costituite da tre differenti parti:

1) Testo
2) Marcatori html
3) Comandi script

In un documento con estensione .asp è consentito utilizzare variabili, cicli, istruzioni di controllo, ecc., grazie alla possibilità di richiamare la sintassi un linguaggio di scripting, come ad esempio il vbscript e il javascript, ma anche perl e rexx. La scelta del linguaggio dipende in primo luogo dalle necessità del programmatore e dal tipo di esecuzione che si vuole avere: se si vogliono eseguire gli script dal lato server è preferibile utilizzare il vbscript, mentre se ci si vuole affidare alla potenza degli "scripting engine" (motore che interpreta i comandi dei linguaggi di scripting e li esegue) dei singoli navigatori è sicuramente meglio utilizzare il javascript, semplice ed efficace.

Prima di iniziare ad imparare la programmazione delle pagine asp tramite un buon numero di esempi commentati, è bene dare un'occhiata preliminare alla sintassi per capire come esso funzioni. Non intendiamo fare qui un elenco completo di tutti i comandi: inizieremo con un insieme di istruzioni base che amplieremo ed illustreremo nel corso delle lezioni utilizzando degli esempi di codice che cerchino di insegnare a creare qualche cosa di realmente utile.

Una prima distinzione che possiamo operare a livello di codice sorgente è a livello di comandi nativi, propri dell'asp, e comandi di scripting che appartengono al particolare linguaggio script utilizzato. Tra i marcatori fondamentali dell'asp ci sono sicuramente i delimitatori, che come nell'html delimitano l'inizio e la fine di una sequenza di codice, e sono rappresentati dai simboli:

```
<% e %>
```

Ad esempio il comando:

```
<% x="ciao" %>
```

assegna alla variabile x il valore "ciao".

Abbiamo già detto che è possibile includere anche script nel codice asp e utilizzare così funzioni create, ad esempio, in javascript o vbscript, richiamandole tramite il comando nativo

```
<% Call _ %>
```

come nell'esempio 1 che mostra come costruire una pagina che visualizzi la data del giorno corrente:

```
ESEMPIO1.ASP:

<% Call PrintDate %>
<SCRIPT LANGUAGE=JScript RUNAT=Server>
function PrintDate()
{
var x
// x è un'istanza dell'oggetto Date
x = new Date()
// il comando Response.Write scrive la data sul navigatore
Response.Write(x.getDate())
}
// Questa è la definizione della procedura PrintDate.
```

```
// Questa procedura manderà la data corrente al navigatore.
</SCRIPT>
```

Dato che abbiamo introdotto il concetto di oggetto, accenniamo brevemente alla sua natura; un oggetto è un'astrazione di una struttura dati, di cui non interessa sapere come è implementata (information hiding), ma quali sono le operazioni possibili su di essa.

L'asp possiede degli oggetti precostituiti, sui quali si può operare con i metodi, che sono delle funzioni intrinseche all'oggetto: per fare un esempio, sull'oggetto automobile un metodo può essere rappresentato dall'operazione di `fare_benzina`, rappresentabile con una notazione del tipo `automobile.fare_benzina`. Sebbene si rassomiglino, bisogna fare attenzione a non confondere i metodi dalle funzioni; queste ultime infatti sono definite dal programmatore e risultano esterne all'oggetto.

Ritornando all'esempio 1, la funzione `PrintDate` definita in javascript è scritta tra i marcatori `<SCRIPT>` e `</SCRIPT>` come sempre, però questa volta sono stati inclusi gli elementi `LANGUAGE=JScript e RUNAT=server.`

Dunque, bisogna specificare obbligatoriamente quale tipo di linguaggio si sta usando e se lo script deve essere eseguito dal alto client o sul server; nel caso non venga specificato il parametro `RUNAT`, il valore predefinito sarà `server`.

Un indubbio vantaggio che deriva dall'uso del delimitatore `RUNAT` di script è costituito dal fatto che il codice sorgente non è mai presente nella pagina html che viene spedita al navigatore dal server. Infatti, il sorgente viene rielaborato dal server che invia come risultato una pagina costruita "al volo" nella quale sono visibili solo i codici html e quelle funzioni per le quali non sia stato specificato il valore "server".

E' interessante notare che non è possibile utilizzare i delimitatori `<% %>` per definire una funzione, dato che non è possibile assegnare nomi a blocchi di codice asp; in seguito vedremo come importare alcune funzioni utilizzando il comando `<!-- #include -->`.

Passiamo ora all'esame dell'oggetto `Response`, il quale consente di gestire l'interazione fra il server e il client. Questo oggetto possiede una serie di metodi che consentono di effettuare una serie di operazioni che avremo occasione do osservare più dettagliatamente nelle varie lezioni di cui si compone questo corso. Ecco di seguito un elenco dei metodi sopra citati:

```
AddHeader
AppendToLog
BinaryWrite
Clear
End
Flush
Redirect
Write
```

Impariamo ora ad usare il metodo

```
Response.Write
```

Tenendo a mente che questo metodo richiede una stringa di testo tra virgolette, o una funzione che restituisca una stringa, esaminiamo l'esempio 2 che illustra come creare e chiamare delle procedure usando due differenti linguaggi di scripting (vbscript e JScript).

```
ESEMPIO 2.ASP :
```

```
<html>
<body>
<table>
<!-- la table non ha i tag <td><tr> ecc. ecc. essi saranno definiti -->
<!-- dalla subroutine Echo definita dallo script in vbscript -->
<!-- cioè il comando <% Call Echo %> chiamerà la subroutine e restituirà -->
<!-- i valori che compileranno la tabella. -->
      <% Call Echo %>
</table>

<!-- stesso discorso soltanto che Print Data restituisce soltanto il valore -->
<!--della data corrente perch, chiama la funzione definita nello script
<!--JavaScript. -->
<% Call PrintDate %>

<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Echo
      Response.Write "<tr><td> Name </td><td>Value </td></tr> "
      ' L'istruzione Set imposta la variabile Params
      Set Params = Request.QueryString
      ' il For itera per ogni stringa inserita nell'URL dopo il punto ?
      '   es.   http://www.italystore.com/esempio2.asp?pippo=3   e   Params(p)   prende   il
parametro 3
      For Each p in Params
            Response.Write " <tr><td>" & p & "</td><td> " & Params(p) & "</TD></TR> "
      Next
End Sub
</SCRIPT>

<SCRIPT LANGUAGE=JScript RUNAT=Server>
function PrintDate()
{
var x
x = new Date()
Response.Write(x.toString())
}
</SCRIPT>
```

Nel precedente esempio abbiamo introdotto la Request, che può essere definito come il primo oggetto che incontriamo ad essere intrinseco al server, dato che esso rappresenta l'elemento di connessione tra il programma client ed il web server. In pratica, esso si occupa di trasmettere le informazioni provenienti da alcune variabili del server (le collection), mentre l'oggetto Response si occupa dell'interazione tra server e client tramite l'utilizzo di metodi quali write, che permette la scrittura a video.

La sintassi propria di questo elemento è:

```
Request[.Collection] ("variabile")
```

Vi possono essere varie collection, ed infatti ne esamineremo alcune nelle prossime lezioni; sempre nell'esempio precedente potete osservare l'utilizzo della collection QueryString, che accetta le stringhe provenienti da un form oppure inserite manualmente, formate da un punto interrogativo seguito da un testo.

Ora, per cominciare, date un'occhiata a questa semplice pagina asp che consente di visualizzare il classico testo "Hello World !" in una pagina html. La particolarità di questo testo consiste nel potere apparire in una dimensione variabile da 3 punti per carattere fino a 7, grazie ad un comando vbscript (For To) che controlla un ciclo di assegnazione di valori alla variabile "i", la quale a sua volta definisce la grandezza del parametro html SIZE; il ciclo viene effettuato su tutto ciò che si trova tra For To e Next.

```
ESEMPIO 3.ASP :
<% For i = 3 To 7 %>
       <font size=<%=i%>>Hello World!</font>
<% Next %>
```

Prima di concludere la lezione, vediamo un altro utilizzo dell'oggetto Request che utilizzando il metodo ServerVariables, permette di richiedere al server una delle variabili di sistema come ad esempio HTTP_USER_AGENT, la quale identifica il nome del navigatore che il client sta usando in quel momento.

```
ESEMPIO 4.ASP :
<% a=Request.ServerVariables("HTTP_USER_AGENT")%>
<% Response.write(a)%>
```

Per concludere questa prima puntata, ricapitoliamo i passi necessari a creare una pagina asp:

1) Aprire un editor di testo (blocco note o Word), o un editor html.
2) Scrivere una pagina che inizi con e si concluda con
3) Utilizzare i comandi <%_%> ed eventuali script
4) Salvare il file con estensione .asp.
5) Effettuare l'upload del file sul vostro sito.
6) Lanciare il navigatore e collegarsi all'URL del vostro file

Per chi usa Windows 95 e vuole esercitarsi sugli asp, la cosa migliore che può fare è procurarsi il Personal Web Server di Microsoft e far girare off-line le proprie applicazioni, dato che le active server pages funzionano solo in presenza di un server web Microsoft.

Abbiamo iniziato con qualche pagina abbastanza semplice, solo per introdurre la sintassi dell'asp e dei linguaggi di scripting ad esso connessi. Certo, conoscere bene il javascript o il vbscript aiuta molto e qualche numero arretrato di inter.net potrà darvi una mano, specialmente grazie al corso di javascript apparso diversi mesi fa. Nel giro di qualche lezione mostreremo qualche linea di codice davvero utile ed interessante, affrontando più approfonditamente le potenzialità di questo linguaggio, fino a discutere di applicazioni di una certa complessità come la costruzione di un form di consultazione e aggiornamento di un database remoto su web.

N.B.: i files .asp vanno lanciati in questa maniera: http://www.pippo.com/prova.asp oppure se si ha istallato il PWS si devono lanciare così: http://127.0.0.1/prova.asp il file prova.asp si deve trovare in c:\intepub\wwwroot\prova.asp.

Per istallare il PWS andate sul CD di Windows 98 e cercate nella cartella ADD-ON\PWS il file setup.exe. I file .asp sono interpretati dal server e non si aprono col browser come per gli html.
Cose interessanti da fare ce ne sono parecchie, ma fino alla prossima puntata vi consigliamo di consultare due siti di indubbio interesse:
http://www.activeserverpages.com
http://aspwire.com

Lezione 2

Creare pagine html interattive non è più un lavoro da professionisti. Oggi, grazie alle active server pages è tutto più semplice e divertente. Impariamo a programmare in asp in poche lezioni

In questa puntata:
come realizzare un feedback form, l'istruzione Select Case, come includere file esterni, come utilizzare le variabili del server ed infine come realizzare un web-counter.

Nella prima puntata si è definito l'Active Server Pages come un filtro ISAPI, aggiunto all'Internet Information Server di Windows NT; in altre parole ASP è un linguaggio di programmazione, di tipo script, che viene eseguito sul server e genera in output del codice HTML.
La differenza dagli altri script languages (Javascript, VBScript), consiste nel fatto che ASP si comporta come un file CGI, cioè non è possibile visualizzare dal browser il codice sorgente del linguaggio, ma solo l'output, inoltre ASP permette anche l'interazione con i DataBase.

In questa puntata si passerà subito all'azione con l'analisi di programmi utili ed immediatamente applicabili, come la costruzione di un feedback form, di cui si può ammirare il listato nell'esempio1

```
Esempio 1:
File HTML del form:

<html>
<head>
      <title>Input form per AspMail</title>
</head>

<body bgcolor="ffffff">
Un semplice modulo di richiesta
<!-- il tag form ha un action che richiama il file aspmail2.asp che processerà le info
provenienti dal modulo -->
<form method="POST" action="aspmail2.asp">
<table border="0">
   <tr>
      <td>Nome</td>
      <!-- ogni box di testo deve avere un nome per essere identificata -->
      <td><input type="TEXT" name="nome" size="25"> </td></tr>
   <tr>
      <td>Telefono #</td>
      <td><input type="TEXT" name="tel" size="15"> </td>
   </tr>
   <tr>
      <td>Indirizzo</td>
      <td><textarea name="indirizzo" rows="3" cols="30"></textarea> </td>
   </tr>
   <tr>
      <td>Email: </td>
      <td><input type="TEXT" name="email" size="20"> </td>
   </tr>
   <tr>
      <td>Eventuali annotazioni</td>
      <td><textarea NAME="commenti" ROWS="4" COLS="30"></textarea> </td>
   </tr>
   <tr>
      <td><input type="SUBMIT" value="Invia"> </td>
   </tr>
</table>
</form>
</body>
```

```
</html>

ASPMAIL2.ASP:
<html>
<head>
<title>Request con le variabili</title>
</head>
<body bgcolor="ffffff">
<!-- request.form("tel") prende il valore contenuto nella box che si chiama "tel" e lo
assegna alla variabile strTel e cosi' per tutti le altre -->

<%
rem il metodo Serve.CreateObject crea delle istanze a oggetti preesistenti come
rem (SMTPsvgMailer) creando cosi' un oggetto.
rem la variabile Mailer è un'istanza dell'oggetto SMTPsvgMailer, cioè è una variabile che
assume rem tutte le caratteristiche ed i metodi dell'oggetto di cui è istanza ovvero il
FromName,
rem FromAddress,il RemoteHost, il Recipient, BodyText, ecc. ecc.

Set Mailer = Server.CreateObject("SMTPsvg.Mailer")
strNome = request.form("nome")
strTel = request.form("tel")
strIndirizzo = request.form("indirizzo")
strEmail = request.form("email")
strCommenti = request.form("commenti")

rem - assegno il nome del mittente e l'indirizzo del mittente
rem - ATTENZIONE: Questi sono campi obbligatori

Mailer.FromName = strNome
Mailer.FromAddress = strEmail

rem - assegno l'indirizzo della macchina SMTP dell'host remoto
rem - grazie alla quale potro' inoltrare la posta.

strMailHost = "mail.xxx.it"
Mailer.RemoteHost = strMailHost

rem - Assegno l'indirizzo del destinatario cioe' il mio.

Mailer.Recipient "nannib@libero.it"

rem - Assegno la stringa dell'oggetto o titolo (Subject) della mail.

Mailer.Subject = "Modulo on line di informazioni"

rem - Inserisco il corpo del messaggio
rem ogni variabile strBody etc. etc. assume il valore della stringa tra le virgolette
piu' i caratteri
rem vbCrLf che sono il Carriage Return ed il Line Feed che servono ad andare a capo.
rem ogni volta alla strBody successiva si concatena il valore della strBody precedente
rem cosi' alla fina la strBody sara' una stringona rappresentante tutto il corpo del
messaggio.

strBody = "Caro Nanni Bassetti" & vbCrLf & vbCrLf
strBody = strBody & "Il mio nome e' " & strNome & vbCrLf
strBody = strBody & "Abito in ." &strIndirizzo & vbCrLf
strBody = strBody & "Il mio numero di telefono e' " & strTel & vbCrLf
strBody = strBody & "La mia E-mail e': " &strEmail & vbCrLf
strBody = strBody & "Ecco i miei commenti al tuo lavoro': " &strCommenti & vbCrLf
Mailer.BodyText = strBody

rem assegno alla variabile strPrimaLinea la stringa "Il tuo Nome e'" seguita dal valore
contenuto rem nella variabile strNome e cosi' per gli altri. Questo per il report finale.

strPrimaLinea = "Il tuo Nome e': " & strNome
strSecLinea = "Il tuo numero di telefono e': " & strTel
strTerzaLinea = "Il tuo indirizzo e': " & strIndirizzo
strQuartaLinea="La tua e-mail e':" &strEmail
strQuintaLinea="Le tue annotazioni sono:"&strCommenti
```

```
rem - Invio il messaggio
rem SendMail può ritornare un valore True o False
rem se Mailer.SendMail (significa che e' True ) allora apparirà sul browser il
rem messaggio "Posta Inviata" ed un report dei dati inseriti altrimenti appare un
messaggio
rem che comunica che ci sono stati problemi nell'invio del messaggio.

if Mailer.SendMail then
Response.Write "Posta Inviata..."

rem infine grazie all'oggetto response, usando il metodo write stamo a video il valore
delle
rem variabili seguite da un

response.write strPrimaLinea & "<br>"
response.write strSecLinea & "<BR>"
response.write strTerzaLinea & "<BR>"
response.write strQuartaLinea & "<BR>"
response.write strQuintaLinea & "<BR>"

else

Response.Write "Mail fallita. Controlla il nome del server e la connessione tcp/ip…<br>"
Response.Write Mailer.Response
end if
%>
</body>
</html>
```

Con questo semplice e comprensibile programmino ora siete in grado di gestire i form on line semplicemente facendo un upload dei files (il file html + il file asp) nella vostra directory di lavoro.
Se si vogliono aumentare o diminuire i campi del form, basterà aumentare o diminuire le variabili associate agli elementi di <input> del modulo.
Chiaramente se si vogliono usare delle check box o dei radio buttons basterà assegnare loro un valore (VALUE="…") ed un nome (NAME="…") come si è sempre fatto in HTML.

Per concludere il commento all'esempio 1, ricordiamo che l'SMTP (Simple Mail Transfer Protocol) è il protocollo di trasmissione usato per la posta elettronica e quando dobbiamo inviare un messaggio dobbiamo sempre indicare l'SMTP server, cioè l'indirizzo della macchina che si occupa di inviare i messaggi.

Prima di buttarsi sulla tastiera bisogna acquisire ancora altri mezzi, infatti per creare un programma in un qualsiasi linguaggio le tre cose fondamentali da sapere sono:

- le istruzioni di assegnazione (es. x=5)
- le istruzioni di iterazione (For…To..Next visto nella scorsa puntata)
- le istruzioni di controllo (If … Then…End If)

Ed a questo proposito ecco qui un esempio ad hoc per ciò che riguarda le istruzioni di controllo:

```
ESEMPIO 2:

<TITLE>Stipendiometro in ASP</TITLE>
<body bgcolor="ffffff">
Inserisci un numero da 0 a 4

<form>
   <p> Inserisci il tuo grado di stipendio (0-4):
   <p><input type="text" name="grado"><br><p><input type="submit" name="Invia">
</form>
```

```
<%
rem creazione della variabile GradoSalario di tipo Variant cioè una variabile che può
contenere
rem diversi tipi di dati (numeri, stringhe, ecc.)

dim GradoSalario

rem assegno il valore della textbox alla variabile GradoSalario

GradoSalario=request.form("grado")

rem l'istruzione Select Case … End Select e' un switch multiplo, cioe' un come se ci
fossero
rem piu' blocchi if then end if

Select Case GradoSalario

   rem nel caso in cui GradoSalario e' uguale a 0 o uguale ad 1 allora stampa "Sei un
   povero barbone"
   rem e cosi' via…
   case 0,1
   response.write("Sei un povero barbone !")
   case 2,3
   response.write("Buongiorno Signore !")
   case 4
   response.write("Eccellenza come va ?")
End Select
%>
</body>
</html>
```

Come promesso nella scorsa puntata, qui vedremo il comando <!-- #INCLUDE della libreria di comandi Server Side Include (SSI) ed analizzeremo gli aspetti più importanti degli oggetti Request e Response.

Al comando #Include, SSI esegue una funzione simile alla direttiva #include del C e C++ e cioè include il contenuto di un file in un file .asp.
Il file incluso può essere di testo, HTML, .asp, di grafica o qualsiasi altro file presente sul server.

Se, per esempio, il vostro sito è composto da centinaia di pagine e si desidera dare a tutte un aspetto uniforme (usando lo stesso template), invece di ricostruire tutto il codice html per ogni pagina, grazie all'SSI basterà inserire il template in un file da includere in tutte le pagine.

L'inserimento di un file esterno ha la seguente sintassi:

```
<!-- #INCLUDE FILE="img.inc" -->
```

dove il file img.inc è posizionato nella stessa directory del file di destinazione può contenere il seguente codice (per esempio):

```
<IMG SRC="http:/www.italystore.com/asp/img/button2.gif" WIDTH="80" HEIGHT="20"
ALT="bottone" ALIGN="CENTER" BORDER="0">
```

così se vogliamo inserire il bottone (button2.gif) in tutto nostro sito web, basterà usare una riga invece di riscrivere tutto il codice html associato.

Un altro metodo per includere i file esterni è:

```
<!-- #INCLUDE VIRTUAL="img.inc" -->
```

dove img.inc si trova nella directory di default: \winnt\system32\inetsrv
però è possibile creare delle sottodirectory, per esempio \winnt\system32\inetsrv\nanni_asp
e l'include diverrà:

```
<!-- #INCLUDE VIRTUAL="/nanni_asp/img.inc" -->
```

Per concludere questa lezione facciamo una breve carrellata sugli oggetti Request e Response;

come avevamo visto l'oggetto Request richiede le informazioni da diverse fonti (le collezioni) e la collezione usata fin'ora è stata la Form, ma ci sono altre come:

la Form, la Cookie, la ServerVariables.

L'insieme ServerVariables informa di tutto ciò che il server sa riguardo al client ed a se stesso, se ad esempio si scrive il seguente codice:

```
<% a=Request.ServerVariables("HTTP_USER_AGENT") %>
<% Response.Write(a) %>
```

si otterrà la visualizzazione su l browser del tipo di software client si sta usando (il che può essere molto utile quando si intende presentare un codice HTML ottimizzato per ogni dato client).
Ecco alcune variabili del server:

HTTP_REFERER URL della pagina indicato dall'utente al documento del sito
HTTP_UA_PIXELS Risoluzione del monitor con il browser del client
HTTP_UA_COLOR Palette dei colori del monitor
HTTP_UA_OS Sistema Operativo del browser dell'utente
HTTP_ACCEPT_LANGUAGE Lingua accettata dal browser
DATE_LOCAL Data corrente nella fascia oraria locale
REMOTE_ADDR IP del client.
ALL_HTTP Tutte le variabili HTTP_*

Dell'oggetto Response abbiamo esaminato solo il metodo Write, quindi è d'obbligo una panoramica su alcuni degli altri metodi:

Response.AppendToLog aggiunge una stringa di non più di 80 caratteri al file di log di IIS.
Response.BinaryWrite permette di visualizzare dei file binari (es. le immagini)
Response.End interrompe l'elaborazione ASP in corso.
Response.Redirect serve per inviare i clients ad un altro URL.

Il mondo degli ASP è sconfinato e qui noi cerchiamo di analizzare i mezzi di più immediato utilizzo e comprensione e con questa nobile missione nel cuore, vi lascio con uno degli strumenti più richiesti ed usati il web-counter degli accessi al vostro sito.

Inserite questo pezzo di codice dove volete nella vostra pagina html, stando attenti a rinominarla *.asp ed il gioco è fatto !

ATTENZIONE: per far funzionare questo procuratevi il file pagecnt dalla mia home page.

```
Counter:

Rem PageCount diventa un'istanza dell'oggetto IISSample.PageCounter che si occupa di
Rem contare gli hits alla pagina
```

```
<%Set PageCount=Server.CreateObject("IISSample.PageCounter")
Rem la variabile Hits prende il valore di PageCount
Hits=PageCount.Hits%>
Rem il tag <%= visualizza il contenuto della variabile Hits.
<%=Hits%>
```

Lezione 3

Metti in linea il tuo catalogo

Impariamo a realizzare un motore di ricerca in grado di interfacciarsi ad un database Access e Sql.

Nelle precedenti lezioni ci siamo avvicinati gradualmente alla sintassi dell'asp, imparando a costruire semplici applicazioni, prendendo confidenza con il linguaggio.
Ora è il momento di passare a qualcosa di più concreto e interessante e quindi proveremo a costruire un motore di ricerca che consenta di interagire con un database remoto via web. Il nostro progetto si articolerà in due sezioni distinte. La prima parte prevedere la creazione di un form (simile al feedback form che abbiamo imparato a creare nell'ultima puntata) attraverso la quale l'utente possa immettere dei dati, mentre in un secondo momento vedremo come implementare un motore asp che processi le informazioni (queries) e le utilizzi per "prelevare" da un database i dati cercati.

Creare l'interfaccia

Cominciamo subito con la scrittura del form che servirà all'utente per interagire con il database, immaginando di avere un negozio di libri online che consenta di cercare un libro in base cercandone i riferimenti in 4 campi: TITOLO, AUTORE, CATEGORIA, CASA EDITRICE.
I primi due campi sono testuali e hanno il vantaggio di consentire ricerche parziali: se ad esempio inseriamo il riferimento MARIO nello spazio dedicato agli autori, il database ci mostrerà tutti i libri il cui autore ha per nome o MARIO; questo, ovviamente, è possibile anche per una ricerca per titolo
Per quanto riguarda gli altri due campi, abbiamo deciso di strutturarli come "menu a tendina" con due opzioni:
E' possibile selezionare la parola chiave "Tutte", che consente di visualizzare tutti i valori associati ad un campo (ad es. tutte le case editrici presenti nel database).
In alternativa, si può scegliere un singolo valore presente in lista. In questo caso, ad esempio, scegliendo "Urania" si otterrà la lista di tutti i libri dell'Urania Editrice.
Diamo subito un'occhiata al codice html che ci consentirà di visualizzare il form appena descritto:

FORM.HTM

```
<HTML>
<HEAD>
   <TITLE>I NOSTRI LIBRI</TITLE>
</HEAD>

<BODY BGCOLOR="#C0C0C0" TEXT="#0F0000" LINK="#0000FF" ALINK="#0000CC" VLINK="#0000FF">
<center>Motore di Ricerca
<form method="post"
   action="motor.asp">
   <p><b>TITOLO</b><br>
   <textarea name="titolo" cols="30"></textarea>
   </p>

   <p><b>AUTORE</b><br>
   <textarea name="autore" cols="30"></textarea>
   </p>

   <p><b>CATEGORIA</b><br>
   <select name="cat">
      <option value="tutte">Tutte</option>
      <option value="DIRITTO">Diritto</option>
      <option value="FANTASCIENZA">
      Fantascienza</option>
   </select>
```

```
    </p>

    <p><b>CASA EDITRICE</b><br>
    <select name="editrice">
        <option value="tutte">Tutte</option>
        <option value="CEDAM">Cedam</option>
        <option value="URANIA">URANIA</option>
        <option value="MONDADORI">
        Mondadori</option>
    </select>
    <br>
    </p>

    <p align="center">
    <input type="submit" name="submit" value="Cerca">
    <input type="reset" name="submit2" value="Reset">
    </p>

</form>
</center>
</BODY>
</HTML>
```

Costruire la base di dati

L'aspetto interessante di questo progetto consiste nel non avere nulla di predefinito: non esistono pagine già pronte che rispondano ad una ricerca dell'utente. Al contrario, il motore in asp è in grado di creare ogni pagina necessaria al momento, interagendo con le informazioni fornite dall'utente attraverso il form.
Dopo l'interfaccia utente è ora il caso di esaminare la base di dati. Un database, in poche parole, è un contenitore nel quale le informazioni possono essere registrate in forma strutturata. Nel nostro caso utilizzeremo una struttura tabellare nella quale alle righe corrispondono i "records" e alle colonne i "campi". Le informazioni relative ad un libro, ad esempio, verranno registrate in un record formato dai campi "Titolo, Autore, Categoria, Casa Editrice, ecc. ecc.".

Il procedimento di creazione di un file database è notevolmente semplificato dagli applicativi moderni che consentono di operare in maniera visuale, diminuendo i tempi di apprendimento e di realizzazione della base di dati. Per ciò che ci proponiamo di fare, Access è lo strumento ideale, semplice ed efficace, ma anche Excel, DB3 possono essere utilizzati senza problemi.
Vediamo in 10 punti come creare un file Access contenente una tabella nella quale registrare i libri:

1) Lanciate Microsoft Access e scegliete "Database Vuoto".
2) Salvate il database dandogli un nome qualsiasi, ad esempio libri.mdb.
3) Cliccate su Tabelle e poi su Nuovo.
4) Scegliete "Visualizzazione foglio dati".
5) Rinominate le colonne con i nomi dei campi (TITOLO, AUTORE, CATEGORIA, CASA EDITRICE, ANNO, PREZZO, COLLANA, IMM, TESTO).
6) Salvate le modifiche e scegliere un nome per la tabella (ad esempio tablibri)
7) A questo punto Access vi chiederà di definire una chiave primaria (nel nostro caso si tratta di un campo contatore che viene incrementato di un'unità per ogni record inserito in tabella), che servirà ad identificare univocamente i record.
8) Riaprite tablibri e cominciare a "riempire" i vari record proprio come se fosse un foglio elettronico di Excel.
9) Salvate ed uscite.

Accendiamo il motore

Ora che abbiamo creato il nostro database (libri.mdb) contenente una sola tabella (tablibri) nella quale sono registrate le informazioni relative ai libri che abbiamo inserito, dobbiamo costruire un motore di ricerca che sappia maneggiare i dati.

A questo proposito, va sottolineato come lavorare sotto Windows NT comporti dei vantaggi non indifferenti. Grazie all'utilizzo di ADO (Activex Data Objects), un componente di Active Server preposto alla gestione dei file (file, directory e database), è possibile infatti accedere a qualsiasi informazione, non importa se questa sia contenuta in un database o in un messaggio di posta elettronica. ADO, infatti, contiene un'interfaccia specifica per la connessione ai database, OLEDB (Object Linking and Embedding per Data Base) la quale attraverso l'interfaccia ODBC (Object Data Base Connectivity) in essa contenuta consente di accedere direttamente a qualsiasi database senza dovere editare delle routine apposite per ogni tipo di dati. Riassumendo, ADO (conosciuto anche come ADODB) è un componente che si occupa dell'accesso "generico" ai dati e contiene al suo interno OLEDB e ODBC, che si occupano di gestire rispettivamente, il primo i database ad oggetti (non ci addentriamo oltre) ed il secondo i database relazionali.
Non ci rimane che analizzare il listato relativo al motore di ricerca vero e proprio:

```
MOTOR.ASP

<HTML>
<HEAD>
   <TITLE>I NOSTRI TITOLI</TITLE>
</HEAD>
<BODY BGCOLOR="#C0C0C0" TEXT="#0F0000" LINK="#0000FF" ALINK="#0000CC" VLINK="#0000FF">
<% rem legge dal form
   titolo = request.form("titolo")
   autore = request.form("autore")
   cat = request.form("cat")
   ed = request.form("editrice")
```

In questo primo blocco di codice osserviamo l'utilizzo dell'oggetto request, tramite il quale i valori provenienti dai diversi campi del database vengono associati a variabili; ad esempio, se l'autore è Pinco Pallo la variabile autore conterrà la stringa "Pinco Pallo".

```
   if titolo = "" then titolo="%"
   if autore = "" then autore="%"
   if cat = "tutte" then cat="%"
   if ed = "tutte" then ed="%"
```

Con questa serie di direttive viene introdotto un controllo che si occupa di sostituire agli eventuali spazi bianchi lasciati nei form, o al valore "tutte", il simbolo % che in linguaggio SQL (Structured Query Language) significa "Qualunque valore del campo".

```
   Sql = "select * from tablibri where titolo like '%" & titolo
   sql = sql & "%' and autore like '%" & autore & "%' and cat like '%" & cat
   sql = sql & "%' and editrice like '%" & ed & "%'"
```

La direttiva che vedete qui sopra può sembrare a prima vista un po' criptica; vediamo di commentarla. Si inizia con l'assegnare alla variabile sql una stringa di comandi SQL il cui significato è:

SELECT (seleziona) * (tutti i campi) FROM (dal) tablibri (la tabella tablibri)
WHERE (nella quale) TITOLO (il campo titolo)
LIKE (è pari a) "%"&titolo&"%" (tutti i valori + il contenuto della variabile titolo+tutti i valori)
AND (e) "%"&autore&"%" ecc. ecc.

Inserire i simboli "%" prima e dopo il contenuto della variabile rende possibile la ricerca anche con dati parziali; se all'interno del form inserisco, ad esempio, nel campo AUTORE solo la lettera M, verranno visualizzati i libri scritti da autori i cui nomi iniziano per M.

```
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
```

Definisco OBJdbConnection come un'istanza dell'oggetto Server.CreateObject("ADODB.Connection").

Quest'oggetto informa il server che si sta creando un mezzo per effettuare una connessione (attraverso ADO) ad un database. Il nome del database è il DSN (Data Source Name, cioè: "nome della sorgente dei dati"), ed è definibile, come amministratore di sistema, da pannello di controllo del server NT. Insomma, si può considerare l'oggetto OBJdbConnection come un telefono, ovvero lo strumento attraverso il quale è possibile effettuare una chiamata (utilizzando il metodo OPEN) al DSN.

```
OBJdbConnection.Open "driver={Microsoft Access Driver(*.mdb)};
dbq=c:\inetpub\wwwroot\newitalystore\libri\libri.mdb"
```

Per meglio comprendere il metodo Open paragoniamolo ad una telefonata mentre per il metodo Execute potremmo pensare ad un utente che da un'estremità della connessioni si accinga ad aprire la bocca per parlare. Sempre riprendendo l'immagine della telefonata, le parole che intercorrono fra i due apparecchi sono i dati contenuti nei record, che vengono estrapolati utilizzando il metodo Open: utilizzando l'oggetto OBJdbConnection il metodo Open apre un canale attraverso il quale può passare il flusso di dati proveniente dalla tabella del database.
Il codice da noi usato serve per impostare una connessione DSN-less (senza-DSN), ideale per chi non è amministratore di sistema, dato che in essa vengono specificati il tipo di driver da utilizzare e la directory in cui risiede il file contenente i dati.
Se avessimo usato un DSN avremmo potuto semplicemente scrivere:

```
OBJdbConnection.Open "libri"
```

dove "libri" è un DSN che indica la posizione del database e quale driver è richiesto per accedere ai dati. Una connessione di questo tipo, però, ha lo svantaggio di dovere richiedere l'intervento dell'amministratore del sistema NT nel momento in cui si decide di spostare il database in un'altra, dato che è necessaria, in questo caso, una modifica del DSN; una connessione DNS-less ha il vantaggio di non richiedere alcuna modifica, dato che la directory di riferimento è già scritta a livello di codice. Un altro sistema che non richiede alcuna conoscenza del server è:

```
OBJdbConnection.Open "driver={Microsoft Access Driver (*.mdb)};dbq=" &
server.mappath("libri.mdb")
```

Però questo sitema "rallenta" un po' il sistema.

```
Set RS=OBJdbConnection.Execute(sql)%>
```

Dopo avere aperto la connessione con l'oggetto OBJdbConnection, si esegue la query contenuta nella variabile sql utilizzando il metodo Execute.
Impostiamo RS come un'istanza dell'oggetto OBJdbConnection; quindi se scriviamo, per esempio, rs("titolo") eseguiamo una query di ricerca sul campo "titolo".

```
<%
response.write ("<center><h2>Libri Trovati
</h2><br><table border=1><tr><td>
```

```
Numero</td><td align=center>Titolo</td>
<td>Autore</td>
<td>Anno</td><td>Categoria</td>
<td align=center>Casa Editrice</td>
<td>Prezzo</td><td>Collana</td>
</tr>") i=0%>
```

L’oggetto response.write stampa a video i comandi html che generano la tabella.

```
<%Do while NOT RS.EOF
i=i+1%>
```

"DO While" "Loop" è un ciclo che dura fino al raggiungimento del RS.EOF, che indica la fine del file (End Of File); per muoversi da un record a quello successivo viene inserita nel ciclo l’istruzione RS.MoveNext.

```
<TR>
<TD><%=i%></TD>
<td><% id=rs("id")
titolo=rs("titolo")%>
<a href="libro.asp?ID=<%=id%>"><%=titolo%></a><br>
</td>
```

Con queste righe decidiamo di mettere in una cella il titolo del libro trovato, recuperandolo da rs("titolo"), che contiene il valore selezionato dall’oggetto RS nel campo "titolo" della tabella tablibri. Da notare, inoltre, che il titolo dell’opera localizzata dalla ricerca apparirà come link ad un file libro.asp che esamineremo nella prossima lezione. Inoltre, per ora, eviteremo di spiegare perché nella variabile "id" abbiamo inserito il valore della chiave primaria (il campo ID contatore) della tabella.

```
<td><%response.write(RS("autore"))
autore=rs("autore")%></td>
<td><%response.write(RS("anno"))
anno=rs("anno")%></td>
<td><%response.write(RS("cat"))
cat=rs("cat")%></td>
<td><%response.write(RS("editrice"))
ed=rs("editrice")%></td>
<td><%response.write("L. "&RS("prezzo"))
prezzo=rs("prezzo")%></td>
<td><%response.write(RS("collana"))
collana=rs("collana")%></td>
</TR>

<%
imm=rs("imm")
RS.MoveNext
Loop
RS.Close
OBJdbConnection.Close%>

</table>
</BODY>
</HTML>
```

Infine arriviamo al metodo Close che si occupa di chiudere tutte le operazioni di esecuzione (Execute) e connessione (Connection) al database; in pratica ricollegandoci all’esempio della telefonata, adesso stiamo salutando e abbassando la cornetta.

Riepiloghiamo:

Si inseriscono delle keyword di ricerca nel form.htm, si preme il tasto CERCA, il form invia le keyword al motore di ricerca motor.asp che costruisce la query (interrogazione) per il database dal quale seleziona i libri con le caratteristiche richieste. Infine, viene generata una pagina che visualizza il titolo, l'autore, la categoria e la casa editrice dei libri trovati. Il titolo però non è un semplice elemento testuale ma un link al file libro.asp. Cosa succede se si clicca su un titolo ?
Lo scopriremo nella prossima puntata…

Lezione 4

Impariamo a creare una pagina web dinamica in asp

Nell'ultima lezione abbiamo visto come costruire un motore di ricerca per una ipotetica libreria, lasciandoci con una domanda: "cosa succede cliccando sul titolo di uno dei libri trovati dal motore?". Prima di concentrarci sul listato dobbiamo capire la logica che sovrintende al funzionamento del tutto; facciamo quindi un breve riassunto

1) Abbiamo un form on line con quattro voci, "Titolo, autore, casa editrice, categoria",nel quale posso scegliere i campi sui quali effettuare la ricerca, presa una decisione va premuto il tasto "CERCA"
2) Premuto il tasto,entra in gioco il motore di ricerca (motor.asp) che legge da un database e seleziona i libri con le caratteristiche specificate nel form.
3) Viene generata una pagina (sempre dal motor.asp) che mostra una lista di libri che possono interessarmi, ma i titoli in realtà sono tutti dei collegamenti ad una pagina chiamata libro,;

rivediamo il link:

```
<a href="libro.asp?ID=<%=id%>"><%=titolo%></a><br>
```

Il fatto che vi sia un parametro "?ID=" significa che quando viene cliccato il titolo di un libro, il browser deve aprire la pagina libro.asp trasmettendo anche il valore della variabile "id", cui è associata il contenuto del campo "ID" del database. Come potete vedere, il valore della variabile id viene inserito in una variabile che convenzionalmente prende il nome di ID ma che potrebbe chiamarsi anche, ad esempio, PIPPO es.

```
<a href="libro.asp?PIPPO=<%=id%>"><%=titolo%></a><br>).
```

Questa operazione consente di richiamare il file libro.asp, passandogli alcune informazioni, relative ad immagini, testi, titolo e altro, che sono contenute nel record relativo al libro selezionato; in base ai dati ricevuti, il motore libro.asp è in grado di generare una pagina html standard,nella quale però alcuni elementi sono presi dal record identificato dalla variabile id.
Facciamo un esempio ipotizzando che la lista di titoli a disposizione del motore di ricerca sia composta da tre libri: L'Isola del tesoro - Giulio Verne - Ed. Adv - ecc. ecc. Il Terzo Gemello - Ken Follett - Ed Mondadori… Neuromante - Gibson - ecc. ecc.
Volendo avere informazioni su "Il Terzo Gemello" basterà cliccare sul link relativo;date un'occhiata alla barra degli indirizzi del navigatore: vedrete comparire la stringa

http://www.ibol.it/libri/libro.asp?ID=2

Ovviamente il valore di ID è uguale a due, dato che il libro in questione è al secondo posto nella lista dei record registrati nel database.. Cliccato il collegamento, infatti, viene costruita una pagina in base ad un modello predefinito, che viene riempito con i dati tratti dal record due. Insomma, con 3 pagine di codice è possibile avere un sito di migliaia di pagine diverse:

- Una pagina per il form on line del motore.
- Una pagina asp che effettua le ricerche nel database e stampa a video la lista dei risultati (come tutti i motori di ricerca su internet).
- Una seconda pagina asp che genera da un modello personalizzato la pagina relativa al libro individuato dal motore.

Ora concentriamoci sul listato:

libro.asp:

```
<HTML>
<HEAD>

<%
REM La variabile ID2 assume il valore della variabile ID presente dopo il "libro.asp?"
REM E' usato l'oggetto request col metodo querystring che serve proprio a "catturare" i
valori
REM delle variabili sulla riga di comando (http://www.ibol.it/libri/libro.asp?ID=2)
id2=request.querystring("id")
REM Definisco la stringa sql utilizzando i comandi dell'omonimo linguaggio in modo che
nel database tutti i campi
REM appartenenti al record numero "ID2"; nel nostro esempio si tratta del record numero 2
sql="select *from tablibri where id="&id2
REM Come abbiamo visto nella lezione precedente, viene effettuata una connessione al
database libri.mdb
REM e tutte le variabili seguenti vengono "caricate" dei valori presenti nel record;
REM nel nostro esempio autore="Ken Follet"

Set OBJdbConnection= Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open                        "driver={Microsoft                        Access
Driver(*.mdb)};dbq="&server.mappath("libri.mdb")
Set RS=OBJdbConnection.Execute(sql)

testo=rs("testo")
titolo=rs("titolo")
autore=rs("autore")
anno=rs("anno")
pagine=rs("pagine")
cat=rs("cat")
ed=rs("editrice")
prezzo=rs("prezzo")
collana=rs("collana")
imm=rs("imm")

REM Qui comincia la costruzione della pagina html Il tag TITLE non è fisso ma visualizza
REM il valore della variabile titolo; nel nostro esempio titolo="Il Terzo Gemello"
%>

<TITLE><%=titolo%></TITLE>
</HEAD>

<BODY BGCOLOR="#ffffff" TEXT="#0F0000" LINK="#0000FF"ALINK="#0000CC" VLINK="#0000FF">

<center>
<h1>Scheda Libro</h1>
<TABLE WIDTH="100%">
   <TR>
   <TD WIDTH="42%">

   <%
   REM In questa cella viene visualizzata un'immagine fissa (Image2.gif) nel caso in cui
   il campo imm del record contenga un "-", altrimenti viene visualizzata un'immagine il
   cui nome viene preso dal
   REM database. Es. imm="Terzgem.gif", che rappresenta la copertina del libro.

   if imm="-"then
      response.write("<IMG SRC=Image2.gif WIDTH=120 HEIGHT=150 BORDER=0>")
   else
      response.write("<IMG SRC="&imm&" WIDTH=120 HEIGHT=150 BORDER=0>")
   end if

   rem Seguono tag html perla barra laterale, i link alle altre pagine del sito e il
   resto. Tutto questo costituisce il modello della pagina libro.asp.%>
```

```
    </TD>

    </TR>

    <TR>
       <TD width="42%" height="22"></TD>
       <TD width="58%" height="22"></TD>
    </TR>

    <TR>
    <TD VALIGN="TOP" ALIGN="LEFT" width="58%">

    <%
    REM Eccoci di nuovo ad una parte dinamica della pagina. Qui impongo che le parole
    Titolo, Pagine,ecc.
    REM vengano visualizzate in grassetto mentre ciò che segue la dicitura è prelevata
    direttamente dal database.
    REM Utilizziamo l'oggetto response col metodo write per stampare avideo i tag html
    insieme alle variabili.

    response.write("<b>Titolo:</b>"&titolo&"<br><b>Autore:
    </b>"&autore&"<br><b>Anno:</b>"&anno&"<br><b>Pagine:
    </b>"&pagine&"<br><b>Editrice:</b>"&editrice&"<br><b>Prezzo:</b>                    L.
    "&prezzo&"<br><b>Categoria:</b>"&cat&"<br><b>Collana:</b> "&collana&"<br><br>")%>
    <center>
    <p> <br>
    <table border=0>
       <tr>
       <td>
       <%
       REM Controllo se la variabile testo è diversa da "-". Questo consente di
       verificare se c'è un testo descrittivo del libro
       REM da importare nella pagina.

       if testo<>"-"then

       REM Il componente File Access permette di accedere a file in formato testo tramite
       due gruppidi oggetti che consentono di trasformare i testi stessi in oggetti
       (TextStream)e di crearne delle istanze (FileSystemObject).
       REM creo un singoloFileSystemObject (objFileSys) e lo uso per creare tutti gli
       oggetti TextStreamnecessari. In questo caso ne serve uno solo.
       Set objFileSys=Server.CreateObject("Scripting.FileSystemObject")
       REM Usiamo il metodoMapPath per specificare il percorso del file da leggere.
       varInputFile=Server.MapPath("/libri")+"\testi\"+testo
       REM Creiamo l'oggettoobjTsIn che tramite il metodo OpenTextFile, apre il file
       individuato dallavariabile (varInputFile).
       Set objTsIn=objFileSys.OpenTextFile(varInputFile)
       REM Qui inizia un ciclo di lettura del file usando la proprietà AtEndOfStream
       dell'oggettoobjTsIn, da usare col metodo ReadLine,
       REM la quale permette di leggere una riga alla volta in un oggetto TextStream.
       REM Il ciclo si ferma quando si raggiunge la fine del file di testo; l'oggetto
       objTsIn viene quindi chiuso col metodo Close.

       Do Until objTsIn.AtEndOfStream
          i=i+1
          response.write(objTsIn.ReadLine)
       loop

       objTsIn.Close

       end if

       %>
       </td>
       </tr>
    </table>
    </center>
</table>
</BODY>
</HTML>
```

Concludiamo l’articolo ricordandovi che per utilizzare il gestore di form apparso nella seconda lezione del corso dovete installare la libreria SMTPSVG.DLL contenuta nel file SMTPSVG.ZIP nel cd rom o scaricabile da internet, mentre per utilizzare il web counter apparso nella seconda lezione bisogna installare la libreria PageCnt.dll presente sul cd rom della terza lezione o scaricabile anch’esso da internet.

Lezione 5

Impariamo a creare una pagina web dinamica in asp

Dopo avere imparato a creare le pagine html contenenti i nostri prodotti è arrivato il momento di mettere mano alla parte "amministrativa" del nostro catalogo in linea. Prima di addentrarci in nuovi argomenti, vediamo di ricapitolare i passi compiuti finora:
Abbiamo creato un form di ricerca contenente i campi Titolo, Autore, Casa editrice, Categoria.
Il tasto "cerca" presente nella form è stato predisposto per inviare i dati inseriti dall'utente ad un file asp (motor.asp), il quale effettua una ricerca all'interno di un database Access (libri.mdb) e visualizza i risultati in una pagina html composta da una tabella con tutti i libri che rispondono ai criteri di ricerca.
Ogni titolo visualizzato nella tabella è stato reso un link ad una pagina asp (libro.asp), che contiene le informazioni relative al libro selezionato, estratte dai campi del record associato al libro stesso.

A questo punto l'interfaccia utente del sito è finita; chiunque può accedere alle nostre pagine e ricercare un libro, estraendo dal database tutte le informazioni necessarie ad identificare l'oggetto ricercato. Ora si tratta di creare, però, gli strumenti per gestire la base dei dati, ovvero inserire, cancellare un libro, oppure modificare una o più informazioni ad esso associate.
Come possiamo fare? In effetti abbiamo a disposizione due possibilità

- Modificare in locale il file contenente i dati e poi caricarlo sulla macchina che gestisce le pagine web.
- Agire sulla base di dati in remoto, utilizzando il navigatore.

La prima soluzione è improponibile dato che rende estremamente laborioso il lavoro di gruppo su uno stesso database e implica notevoli trasferimenti da e verso il server, a meno che questo non sia direttamente raggiungibile in locale.
La seconda soluzione, invece, permette a più operatori, anche distanti migliaia di chilometri tra loro, di lavorare sul database e inoltre non obbliga a scaricare tutto il file di database anche quando si è modificato un solo record. Prenderemo quindi in esame la seconda soluzione, creando gli strumenti appropriati per gestire in remoto il database del sito che abbiamo appena creato.

La prima cosa da fare è creare un form html che ci consenta di proteggere da sguardi indiscreti la parte del sito dedicata alla gestione; è ovvio che non sarebbe una soluzione ideale lasciare libero accesso alla modifica dei dati. In pratica, il form preleva la password dell'utente e la invia ad un file asp (pass.asp), che la confronta con quella contenuta in un database di codici. In caso di responso positivo, il navigatore potrà accedere alle pagine html più riservate, altrimenti viene rediretto ad una pagina di default..

Vediamo il codice del file pass.asp:

PASS.ASP:

```
<%
rem Definisco una variabile pass
dim pass

rem Inserisco nella variabile passw la password scritta nel form html
pw=request.form("passw")

rem leggo tutti i record della tabella pass del database pass.mdb
sql="select * from pass"
```

```
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open              "driver={Microsoft              Access              Driver
(*.mdb)};dbq=c:\inetpub\wwwroot\domini\italystore\annunci\pass.mdb"
Set RS=OBJdbConnection.Execute(sql)

rem Carico nella variabile pw2 il valore del campo pw del record del datatbase pass.mdb
pw2=rs("pw")

rem Effettuo il confronto tra la password presente nel db e la password inserita nel form
if pw<>pw2 then

rem Se le due password sono diverse allora ritorno alla home page
response.redirect "default.asp"

rem Altrimenti inserisco nella variabile pass la stringa "ok"
else
pass="ok"
%>

rem A questo punto inserisco il file modifica.asp
<!-- #include file="modifiche.asp" -->
<% end if %>
```

Se le password sono uguali, l'utente può accedere al file modifica.asp che permette di scegliere se aggiungere o cancellare un record del database.

MODIFICHE.ASP:

```
<%
passb=request.form("pass")
rem  Controllo  se  la  variabile  pass  ha  il  valore  "ok".  La  variabile  pass  viene
direttamente dal
rem file pass.asp.
if (pass="ok") then %>

<html>
<HEAD>
       <TITLE>Il Database</TITLE>
</HEAD>

<BODY    background="sfondo.gif"    BGCOLOR="#C0C0C0"    TEXT="#0F0000"    LINK="#0000FF"
ALINK="#0000CC" VLINK="#0000FF"><p>

rem Da qui posso scegliere se aggiungere o cancellare o aggiornare un record, la mia
scelta rem verrà trasmessa al file
rem scelta.asp
<form method=post action="scelta.asp">
   <center>
   <input type=submit name="aggiungi"value="Aggiungi">
   <input type=submit name="cancella" value="Cancella">
   <input type=submit name="cambia" value="Cambia Password">
   <input type=hidden name="pass" value="ok">
   </center>
</form>
</BODY>
</HTML>

<%
rem Se pass è diverso da "ok" allora torno alla home page default.asp

else
response.redirect "index.htm"
end if%>
```

Fin qui nessun problema: tutto concerne la verifica della password,ma una volta passata questa fase possiamo concentrarci sul codice che si occupa delle varie operazioni sul database.

SCELTA.ASP:

```
<%
pass=request.form("pass")
s=request.form
rem Se si sceglie di aggiungere un record allora entra in azione questa sezione del
programma, che approfondiremo nelle prossime lezioni
if s="aggiungi=Aggiungi&pass=ok" then
%>

<html>
<HEAD>
   <TITLE>Il Database</TITLE>
</HEAD>

<BODY    background="sfondo.gif"    BGCOLOR="#C0C0C0"    TEXT="#0F0000"    LINK="#0000FF"
ALINK="#0000CC" VLINK="#0000FF">

<form method=post action="add.asp">
   <center><h2>Aggiungi Libro</h2><br>
   <table border=1>
   <tr>
      <td align=center>Titolo</td><td align=center>Autore</td>
      <td align=center>Anno</td></tr>
   <tr>
   <TD><input type=text name="titolo"></TD>
   <td><input type=text name="autore"></td>
   <td><input type=text name="anno"></td>
   </tr>
   <tr>
      <td align=center>Categoria</td><td align=center>Casa Editrice</td>
      <td align=center>Prezzo</td>
   <tr>
      <td><input type=text name="cat"></td>
      <td><input type=text name="ed"></td>
      <td><input type=text name="prezzo"></td>
   </tr>
   <tr>
      <td align=center>Collana</td><td align=center>Testo</td>
      <td align=center>Foto</td>
   </tr>
   <tr>
      <td><input type=text name="collana"></td>
      <td><input type=text name="testo"></td>
      <td><input type=text name="imm"></td>
   </tr>
   <tr>
      <td align=center>Pagine</td>
   </tr>
   <tr>
      <td><input type=text name="pagine"></td>
   </TR>
   </table>
   <br>
   <input type=hidden name="pass" value="ok">
   <input type=submit name="ok" value="Aggiungi">
</form>
</BODY>
</HTML>
<%end if
rem Se s="cancella=Cancella&pass=ok" significa che abbiamo scelto l'opzione CANCELLA e
rem che siamo autorizzati ad usarla

if s="cancella=Cancella&pass=ok" then

rem A questo punto creiamo un motore di ricerca che trovi solo il record che ci interessa
rem cancellare, altrimenti dovremmo visualizzare tutta la base di dati…
%>

<html>
<head>
```

```
    <title>Modifica- by Italystore.com</title>
</head>

<body bgcolor="#FfFfff">
<center>
<!—Qui vediamo apparire una casella di testo da riempire col titolo del libro che ci
interessa cancellare; ovviamente, se non ricordiamo tutto il titolo possiamo inserire
anche solo alcune parole contenute in esso. Dopo la pressione del tasto CERCA viene
eseguito il file cancscel.asp che effettua una ricerca per titolo nel database e
visualizza i risultati a video. Chiaramente la ricerca può essere effettuata su più campi
ma, per semplificare, ora utilizziamo solo il campo titolo-->

<form method="post" action="cancscel.asp">
    <p><br>
    <b>Cerca il titolo del libro da cancellare:</b>
    <input type="text" name="tit"><br>
    <!— Con input type=hidden inviamo il valore della variabile pass alla pagina chiamata
    dal form, ovvero cancscel.asp -->
    <input type=hidden name="pass" value=<%=pass%>>
    <input type="submit" name="vai" value="Cerca">
</form>

</center>
</body>
</html>

<%end if
rem Diamo ora la possibilità di cambiare la password da web.

if s="cambia=Cambia+Password&pass=ok" then
sql="select * from tablibri"
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open              "driver={Microsoft              Access              Driver
(*.mdb)};dbq=c:\inetpub\wwwroot\domini\italystore\libri\libri.mdb"
Set RS=OBJdbConnection.Execute(sql)%>

<html>
<head>
    <title>Modifica- by Italystore.com</title>
</head>

<body bgcolor="#FfFfff">
<br>
<!-- Come al solito c’è un piccolo form che manda la nuova password al file update.asp,
insieme alla variabile pass che contiene il valore "ok". In questo modo si evita che
qualcuno si colleghi direttamente alla pagina update.asp senza autorizzazione.-->
<form method="post" action="update.asp?id=<%=rs("id")%>">
Password:
<input type=text name="upw"><br>
<input type=hidden name="pass" value="ok">
<input type=submit value="Aggiorna">
</form>
</body>
</html>
<%end if
%>
```

Ciò che abbiamo fatto finora non è molto difficile; anzi si poteva evitare chiedendo all’amministratore di sistema di proteggere le pagine tramite una password da server. La soluzione scelta, però, ha il vantaggio di rendere indipendente il sito dall’amministratore e di rimettere nelle vostre mani la decisione su chi debba accedere alle pagine rilevanti, consentendovi inoltre di modificare i parametri in tempo reale.
Ora passiamo a dare un’occhiata da vicino al file canc.asp che si occupa di cancellare i record dal database; nelle prossime lezioni approfondiremo le altre operazioni rese possibili da questo file.
Prima, però, diamo uno sguardo al cancscel.asp:

CANCSCEL.ASP:

```
<html>
<head>
<title>Modifica- by Italystore.com</title>
</head>
<body bgcolor="#FfFfff">
<%
rem Solita richiesta della variabile pass per verificare che siamo autorizzati ad eseguire
rem questo script
pass=request.form("pass")
if pass="ok" then
rem Viene passato anche il valore di tit e inserito in una variabile chiamata tit
tit=request.form("tit")
rem Si analizza il contenuto di tit: se essa è vuota allora assumerà il valore "%", che significa rem "tutti" e consente di visualizzare il contenuto di tutta la base dati.
if tit="" then tit="%"
sql="select * from tablibri where titolo like '%"&tit&"%'"
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "driver={Microsoft Access Driver (*.mdb)};dbq=c:\inetpub\wwwroot\domini\italystore\libri\libri.mdb"
Set RS=OBJdbConnection.Execute(sql)%>
<%
response.write ("<center><h2>Articoli Presenti</h2><br><br><table border=1><tr><td>Art. da Cancellare</td><td>Autore</td><td align=center>Titolo</td><td>Editrice</td><td>Prezzo</td><td>Cat.</td><td align=center>Anno</td></tr>") %>
<%Do while NOT RS.EOF%>
<TR>
<td>
<!—Nella prima cella della tabella inseriamo un piccolo form che chiama il file canc.asp passandogli il valore, inserito nella variabile c, del campo ID del record. Insomma, indichiamo quale dei record deve essere cancellato. Nella prima cella apparirà un tasto con etichetta "Cancella" il quale, se premuto, effettuerà la cancellazione del record. -->
<form method=post action="canc.asp?c=<%=RS("ID")%>">
<input type="hidden" name="pass" value="<%=pass%>">
<input type="hidden" name="tit" value="<%=tit%>">
<input type="submit" name="Cancella" value="Cancella">
</form>
</td>
<TD><%=RS("autore")%></TD>
<td><%=RS("titolo")%></td>
<td><%=RS("editrice")%></td>
<td>
<%=RS("prezzo")%>
</td>
<td><%=RS("cat")%></td>
<td><%=RS("anno")%></td>
</TR>
<%RS.MoveNext
Loop
RS.Close
OBJdbConnection.Close
%>
</table><br>
<form method=post action="modifiche.asp">
<input type=hidden name="pass" value="ok">
<input type=submit name="back" value="Torna al menu di gestione">
</form>
<br>
</center>
</body>
</html>
<%else
rem Nel caso in cui il valore di pass sia diverso da "ok", significa che stiamo tentando di usare rem il file cancscel.asp senza essere passati attraverso la richiesta di password iniziale.
response.write "<h1>Accesso Negato !</h1>"
end if
```

```
%>
```
Adesso abbiamo a video una tabella di uno o più libri, contenete nella prima colonna un tasto con etichetta "CANCELLA"; se premiamo il pulsante eseguiremo il file canc.asp, che eliminerà il record corrispondente al libro.
CANC.ASP:
```
<%
rem Come sempre prendiamo i valori di tit e di pass dal file precedente cancscel.asp
tit=request.form("tit")
pass=request.form("pass")
rem Solita verifica di autorizzazione
if pass="ok" then
rem La variabile canc prende il suo valore dalla variabile c sulla stringa di comando.
rem C contiene il valore del campo contatore ID che identifica univocamente il record nel
rem database
canc=request.querystring("c")
%>
<html>
<head>
<title>Cancella - by Italystore.com</title>
<body bgcolor="#FfFfff">
<%
rem Cancella i record selezionati
Set OBJdbConnection2 = Server.CreateObject("ADODB.Connection")
OBJdbConnection2.Open "driver={Microsoft Access Driver (*.mdb)};dbq=c:\inetpub\wwwroot\domini\italystore\libri\libri.mdb"
rem L'unica rispetto al motore di ricerca è nella stringa di comandi sql. Infatti, al posto
rem di "Select" abbiamo "Delete", che significa "cancella". L'espressione sql tradotta
rem in italiano significa quindi: "Cancella tutti i campi della tabella tablibri in cui il campo
rem contatore ID è uguale al valore contenuto nella variabile canc"
sql="Delete * from tablibri where ID="&canc
Set RS2=OBJdbConnection2.Execute(sql)
OBJdbConnection2.close
rem Dopo l'operazione di cancellazione viene visualizzato un messaggio esplicativo e viene
rem data la possibilità di tornare indietro alla pagina cancsel.asp tramite un form.
rem Abbiamo scelto di usare un form e non un link perché in questo modo possiamo passare
i rem valori di pass e tit al file cancscel.asp.
%>
<H2>CANCELLAZIONE EFFETTUATA!</H2>
<br>
<form method=post action="cancscel.asp">
<input type="hidden" name="tit" value="<%=tit%>">
<input type="hidden" name="pass" value="ok">
<input type="submit" name="goon" value="Continua a cancellare">
</form>
</body>
</html>
<% else
rem Se pass è diverso da "ok"
response.write "<h1>Non sei autorizzato a cancellare !</h1>"
end if %>
```

Siamo arrivati alla fine. Vi sembra tutto un po' lungo e complicato? A dire il vero, l'unica parte davvero ostica è rappresentata dalle pagine di raccordo tra le scelte da effettuare (password,cancella, aggiorna, ecc.) e l'esecuzione dei file asp Il resto è tutto sommato semplice: il vero strumento per cancellare i record è costituito dai due file cancel.asp e canc.asp; inoltre quest'ultimo file è una versione modificata del motori di ricerca, già visto nelle precedenti lezioni.
Ora è tempo di concludere questa puntata e lasciarvi sperimentare il nuovo codice che abbiamo appena esaminato. Nel prossimo numero vedremo come inserire nuovi record nel database e impareremo qualche altro comando asp. Alla prossima allora!

Asp, questi sconosciuti – Corso di asp parte VI
Impariamo a creare una pagina web dinamica in asp
di Nanni Bassetti nannib@libero.it

Continua il nostro viaggio nell'affascinante mondo degli ASP (Active Server Pages) di Microsoft.
Nell'ultima lezione abbiamo visto come cancellare dei records di un database residente sul server, tramite web, oggi vedremo come aggiornare ed aggiungere dei record.
Con quest'ultime due funzioni abbiamo completato l'insieme dei mezzi per gestire un database remoto, sarà compito vostro creare una pagina dalla quale poter scegliere che tipo di operazioni fare.
Naturalmente questa pagina sarà accessibile solo ai possessori di password autorizzata, altrimenti chiunque potrebbe commettere chissà quali nefandezze sulla vostra base di dati.
La scorsa puntata abbiamo visto che una volta immessa una password nel form in html, essa viene confrontata con quella presente nel database della password, se le due combaciano allora si entra nella pagina dalla quale si possono compiere le operazioni di cancellazione ed aggiunta di record nel database dei libri (libri.mdb) altrimenti si torna all' home page.
Vediamo come si aggiorna un record, nello specifico vediamo come facciamo a cambiare la nostra password on-line.
Se dalla pagina modifiche.asp clicchiamo su AGGIORNA PASSWORD, allora viene lanciato il file Update.asp:

```
Update.asp
<%
REM solita procedura di controllo della variabile pass per vedere se contiene il valore
"ok" altrimenti non si esegue il codice di update.asp
pass=request.form("pass")
if pass="ok" then%>
<HTML>
<HEAD>
<TITLE>Agiiorna il Database</TITLE>
</HEAD>
<BODY    background="sfondo.gif"    BGCOLOR="#C0C0C0"    TEXT="#0F0000"    LINK="#0000FF"
ALINK="#0000CC" VLINK="#0000FF">
<%
REM la upw assume il valore del campo di tipo text col name="upw" che sarebbe la nuova
password che desidero usare
upw=request.form("upw")
rem solita procedura di connessione al database di nome pass.mdb.
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open             "driver={Microsoft             Access             Driver
(*.mdb)};dbq=c:\inetpub\wwwroot\domini\italystore\libri\pass.mdb"
REM qui c'è la novità l'istruzione sql (Structured Query Language) Update, che serve
appunto ad aggiornare un record.
sql="update pass set pw='"&upw&"' where id="&request.querystring("id")
REM commentiamola: "Aggiorna la tabella pass (del db pass.mdb) poni il campo pw uguale al
valore contenuto in upw dove il campo id è uguale al valore della variabile id mandata
sulla stringa di comando dal form.
Set RS=OBJdbConnection.Execute(sql)
rem esegue l'istruzione sql
OBJdbConnection.Close
%>
<center>
<h2>Database Password Aggiornato !</h2>
premi back per tornare indietro !
</CENTER>
</BODY>
</HTML>
<%
REM se pass fosse stato diverso da "ok" allora veniva fuori questa scritta !
else
response.write "<h1>Accesso Negato !</h1>"
end if%>
```

```
Naturalmente il sistema di "aggiornamento" del record può essere esteso anche a più campi
e a più records, ad esempio potremmo aggiornare i campi prezzo e quantità dei record di
un database di articoli del nostro magazzino.
Ma oltre l'aggiornamento il sistema di gestione remota di un database deve poter anche
aggiungere dei records e allora:
Add.asp
<%
pass=request.form("pass")
if pass="ok" then%>
<HTML>
<HEAD>
<TITLE>Il Database</TITLE>
</HEAD>
<BODY    background="sfondo.gif"    BGCOLOR="#C0C0C0"    TEXT="#0F0000"    LINK="#0000FF"
ALINK="#0000CC" VLINK="#0000FF">
<%
REM solito sistema per caricare i valori delle variabili inviate dal form.
titolo=request.form("titolo")
autore=request.form("autore")
cat=request.form("cat")
ed=request.form("ed")
anno=request.form("anno")
prezzo=request.form("prezzo")
collana=request.form("collana")
pagine=request.form("pagine")
testo=request.form("testo")
imm=request.form("imm")
REM una serie di if che controllano se i campi sono vuoti, per esempio di un libro
potremmo avere solo il titolo e l'autore a questo punto gli altri campi devono assumere
dei valori fittizi come il "-" o lo "0" (per quelli numerici)
if titolo="" then titolo="-"
if autore="" then autore="-"
if ed="" then ed="-"
if cat="" then cat="-"
if anno="" then anno=0
if prezzo="" then prezzo=0
if collana="" then collana="-"
if pagine="" then pagine="-"
if imm="" then imm="-"
if testo="" then testo="-"
```

rem solita procedura di connessione al database libri.mdb
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "driver={Microsoft Access Driver (*.mdb)};dbq=c:\inetpub\wwwroot\domini\italystore\libri\libri.mdb"
REM qui c'è la novità, infatti pongo la variabile RS come oggetto Recordset.
L'oggetto Recordset si usa per creare un set di records, appunto un insieme di record sul quale lavorare.
L'oggetto Recordset è "l'officina" di ADO (Active Data Object) (ne abbiamo parlato nelle scorse lezioni), cioè il posto in cui sono eseguite quasi tutte le attività di gestione del database.
RS diventa un'istanza dell'oggetto Recordset quindi può usufruire del metodo Open.
Open apre la tabella "tablibri", sulla connessione OBJdbConnection (definita in precedenza), i numeri 3,3 che seguono rappresentano rispettivamente:
il CursorType
il LockType
Il CursorType definisce il tipo di "cursore", il quale è una rappresentazione dei dati (records) e definisce anche il tipo di viste possibili su di essi.
I tipi di cursori sono di 4 tipi:
Cursore di default valore:0 è come un cursore statico ad eccezione del fatto che permette solo spostamenti in avanti e non indietro, poiché questo cursore non deve tener traccia dei record aggiunti, modificati o cancellati da atri utenti.

Cursore Keyset valore: 1 possono vedere le modifiche apportate da altri utenti, nonché spostarsi tra i record in avanti ed indietro. Non possono vedere i record aggiunti o cancellati da gli altri utenti.
Cursori Dinamici valore:2 possono vedere qualsiasi cosa: modifiche, aggiunte e cancellazioni fatte da altri utenti. Supportano, inoltre tutti gli spostamenti.
Cursori Statici valore: 3 permettono gli spostamenti in avanti ed indietro. Non sono in grado di rilevare le modifiche ai dati apportate da altri utenti.
Il valore di LockType (o di BLOCCO) determina che tipo di blocco il fornitore di dati DBMS (Data Base Management System), ossia il database realizzato con Access (libri.mdb), deve usare quando si apre un Recordset, questo ai fini del controllo di concorrenza sugli accessi ai dati da parte di più utenti contemporaneamente.
Per definizione le applicazioni client/server implicano che più di una persona può accedere contemporaneamente ad un database, questa caratteristica è chiamata concorrenza.
Per un accesso concorrente ci deve essere un deposito al quale più utenti accedano per leggere e cambiare i dati.
Ciò significa che il DBMS deve garantire l'accuratezza dei dati memorizzati e per gestire in modo efficiente un accesso concorrente ai dati, è necessario fronteggiare un altro problema, che è chiamato interferenza.
L'interferenza è quel fenomeno che accade quando due fenomeni della stessa natura si sovrappongono causando un disturbo reciproco, questo principio è facilmente estensibile all'azione di due utenti che stiano compiendo la stessa operazione sullo stesso record di un database.
Per evitare l'interferenza esistono le opzioni di blocco:
adLockReadOnly valore: 1 i dati si possono soltanto leggere.
AdLockPessimistic valore: 2 i dati sono bloccati appena qualcuno comincia ad effettuare operazioni di modifica, così da mantenere un'integrità assoluta dei dati, ma con spiacevoli rallentamenti del sistema, infatti finché l'utente non finisce le sue modifiche quei dati sono bloccati. Il blocco pessimistico ha il vantaggio di essere molto sicuro, specialmente su sistemi come Internet, che ha la possibilità di far accedere tantissimi utenti contemporaneamente, ma ha lo svantaggio di creare parecchie difficoltà su Internet proprio perché qualche utente potrebbe andare a pranzo e lasciare i records bloccati, oppure i rallentamenti della rete causerebbero un blocco abbastanza lungo nel tempo ecc. ecc.
AdLockOptimistic valore: 3 il blocco ottimistico crea un buffer temporaneo in cui vengono conservati gli aggiornamenti e le modifiche sui dati, mentre i dati originali sono ancora accessibili agli altri utenti, quando si lancia un comando di aggiornamento (Update) allora i dati vengono bloccati e appena finito l'aggiornamento il blocco viene rilasciato.

```
Set RS=Server.CreateObject("ADODB.Recordset")
RS.Open "tablibri",OBJdbConnection,3,3
REM Usiamo il metodo AddNew per immettere il valore della variabile titolo nel campo "titolo" del database, il valore della variabile autore nel campo "autore", e così via per tutti gli altri.
RS.AddNew
rs("titolo")=titolo
rs("autore")=autore
rs("cat")=cat
rs("editrice")=ed
rs("anno")=anno
rs("prezzo")=prezzo
rs("collana")=collana
rs("pagine")=pagine
rs("testo")=testo
rs("imm")=imm
REM dopo aver caricato i valori delle variabili lanciamo il metodo Update (dato che abbiamo usato il blocco ottimistico) che serve a bloccare i dati fino a quando non finisce l'aggiornamento.
rs.update
REM a questo punto chiudiamo la connessione col database e l'apertura del recordset.
RS.Close
```

```
OBJdbConnection.Close
%>
<center>
<h2>Database Aggiornato !</h2>
premi back per tornare indietro !
</CENTER>
</BODY>
</HTML>
<% else
response.write "<h1>Accesso Negato!</h1>"
end if%>
```

Riepilogando in queste lezioni abbiamo visto:
I comandi basilari dell'ASP (Active Server Pages)
Un gestore di feedback form.
La realizzazione di un motore di ricerca.
La costruzione dinamica di una pagina html
Come cancellare dei records da un database via web
L'aggiornamento e l'aggiunta dei dati di un database via web.
Utilizzando questi mezzi si possono inventare e implementare tantissime applicazioni utili e carine, senza spendere !
Potete realizzare motori di ricerca, shopping cart per il commercio elettronico, bacheche su web gestite automaticamente, web chat, ecc. ecc.
Alcuni esempi (realizzati dal sottoscritto):

http://www.italystore.com/annunci (inserimento e lettura di annunci economici)
http://www.giroscopio.com/racconti (inserimento e lettura di racconti)
http://www.giroscopio.com (motore di ricerca)
http://www.italystore.com (shopping cart)

I mezzi sono quelli che vi ho dato manca ancora un ultimo ingrediente per realizzare quasi tutto: session.sessionId.
L'oggetto session con il suo metodo sessionId permette di assegnare ad ogni browser un numero univoco (come un cookie), quindi se in un listato asp scrivo:
<% x=session.sessionid %>
e poi vado a scrivere il valore della variabile x in un campo (es. il campo "user") di un database, ovviamente potrò "personalizzare" alcuni records di quel database.
Questo è molto utile ad esempio nella realizzazione di uno shopping cart (carrello per gli acquisti elettronici), infatti quando un cliente sta leggendo dal database i suoi acquisti non sta facendo altro che leggere tutti i records che hanno nel campo "user" il suo numero session.sessionid.
È intuitivo che se ci sono due clienti contemporaneamente uno non vedrà gli acquisti dell'altro proprio perché ognuno leggerà i suoi records personali.
La stringa sql sarà: "select * from tabella_acquisti where user="&x
e cioè leggi dalla tabella_acquisti del database tutti i records che hanno il valore di x nel campo user.
Facile vero ? No ? Bè con un po' d'impegno lo diventerà….ah chiaramente vi consiglio l'acquisto di un buon libro sugli asp.
Buona Fortuna e buon lavoro !