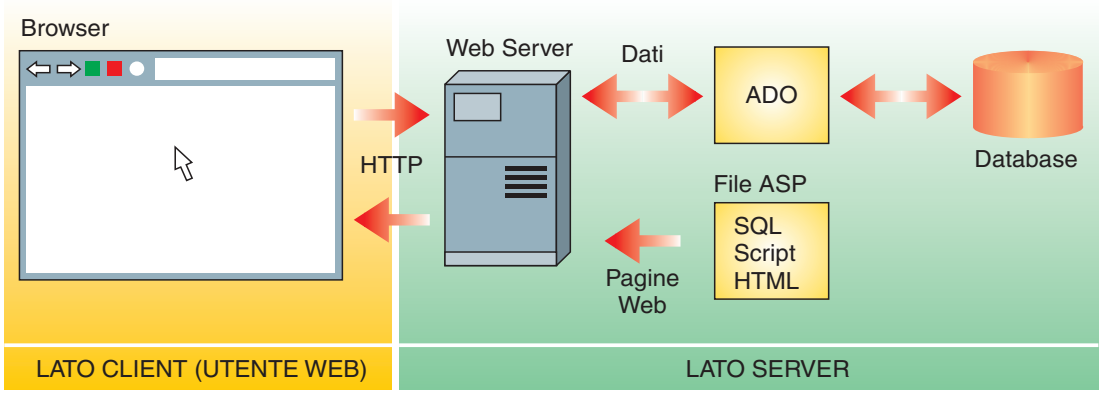
3. Programmazione con le pagine ASP classiche

1. Le pagine ASP in generale

ASP (*Active Server Pages*) è una tecnologia Microsoft che consente di scrivere codice eseguibile (**script**) lato server, inserendo il codice all'interno delle pagine Web.
Per eseguire le pagine ASP occorre quindi disporre di un server Web che sia in grado di interpretare gli script in formato ASP: principalmente il Web server di Microsoft che si chiama **IIS** (*Internet Information Services*).
Una pagina ASP è un normale file di testo, salvato con l'estensione **.asp**, che può contenere testo, codici HTML, o script scritti con altri linguaggi, per esempio *JavaScript*.
La directory principale (*home directory*) del server Web è denominata di solito **\Inetpub\wwwroot**. È buona norma registrare le pagine ASP in un'altra sottodirectory di *Inetpub*, avente di solito nome **Scripts**, sulla quale sia stato stabilito il permesso di esecuzione degli script.
La pagina ASP è eseguita sul server in risposta a una richiesta proveniente dal browser dell'utente, in genere quando l'utente fa clic con il mouse su un pulsante di comando contenuto in un form HTML. In sostanza, l'utente, da una pagina HTML, richiede di eseguire l'elaborazione contenuta nello script ASP; il server Web esegue le istruzioni contenute nella pagina ASP riga per riga e restituisce i risultati sotto forma di una pagina HTML che viene costruita dinamicamente e inviata al browser dell'utente.



Questo significa anche che l'utente finale che utilizza il browser non può vedere il sorgente della pagina ASP residente sul server, ma solo la pagina HTML generata dallo script eseguito dal server.
Lo sviluppatore di pagine ASP può verificare il funzionamento degli script sulla propria macchina, prima di pubblicarli in Internet, utilizzando un server Web installato localmente sul suo computer,

Il linguaggio di default utilizzato per scrivere il codice delle pagine ASP è il linguaggio **VBScript**, che si chiama così perché usa sintassi e funzioni appartenenti al linguaggio *Visual Basic.*
In alternativa, si può utilizzare anche il linguaggio **JScript**, che è la versione Microsoft di *JavaScript*, oppure il linguaggio **C#** (si legge *C sharp*), un linguaggio a oggetti, anch'esso di Microsoft.

I linguaggi di scripting sono di solito interpretati, quindi, poiché il server Web traduce le istruzioni una di seguito all'altra durante l'esecuzione, un eventuale errore riscontrato nel codice provoca l'arresto dell'esecuzione e l'invio di una pagina HTML al browser dell'utente contenente il messaggio di errore. Il programmatore di pagine ASP ha tuttavia la possibilità di gestire le situazioni di errore, utilizzando le funzioni che intercettano l'errore.

L'applicazione più comune delle pagine ASP riguarda la possibilità di accedere ai dati contenuti nelle tabelle di un database residente sul server. L'utente può inserire, modificare, cancellare i dati delle tabelle, oppure eseguire interrogazioni e comandi espressi in linguaggio SQL, semplicemente utilizzando il browser. Per questo tipo di applicazione si utilizza la tecnologia **ADO** (*ActiveX Data Objects*), la libreria software standard di Microsoft per l'accesso ai database. La recente evoluzione delle pagine ASP ha prodotto la versione **ASP.NET** per le pagine e **ADO.NET** per l'accesso ai database: queste tecnologie saranno presentate nei paragrafi successivi, dopo la trattazione delle pagine ASP classiche.

2. Il linguaggio di scripting per le pagine ASP

Come già detto in precedenza, una pagina ASP è un normale file di testo, salvato con l'estensione **.asp**: esso contiene parti delimitate dai codici HTML (**tag**) e parti di codice ASP (**script**) delimitate da una coppia di simboli

```
<% ... %>
```

Il codice può anche essere suddiviso in più parti all'interno della stessa pagina Web.
In mancanza di dichiarazione contraria, per default, il linguaggio di scripting è *VBScript*.
Per impostare il linguaggio di scripting si deve aggiungere, all'inizio della pagina ASP, l'istruzione

```
<%@ LANGUAGE = VBScript %>
```

Un primo semplice esempio mostra come si possa inserire il codice all'interno della pagina ASP, insieme ai tag HTML.

Visualizzare la data e l'ora corrente con una pagina Web.

La funzione del linguaggio *VBScript* che restituisce la data e l'ora corrente è **Date()**.
L'**operazione di output** è rappresentata dal segno **=** seguito dalla variabile o dalla funzione di cui si vuole visualizzare il valore.

PAGINA ASP (*Dataora.asp*)

```
<%@ LANGUAGE = VBScript %>
<HTML>
<HEAD>
<TITLE>Data e ora</TITLE>
</HEAD>
<BODY>
<CENTER>
<H2>Data e ora del sistema</H2>
<P><% =Date() %></P>
</CENTER>
</BODY>
</HTML>
```

Il testo dello script viene salvato nella sottodirectory del server Web, per esempio in *\Inetpub\Scripts*, con il nome *Dataora.asp*.

Scrivendo poi nella casella dell'indirizzo del browser

```
http://localhost/Scripts/Dataora.asp
```

si ottiene una pagina Web con il titolo, la data e l'ora del sistema. Aprendo dal browser la finestra del sorgente HTML (menu *Visualizza*, scelta *HTML*), si può notare che non viene visualizzata la pagina ASP, ma la pagina HTML generata dinamicamente dallo script eseguito dal server.

Per documentare le pagine ASP, il programmatore può utilizzare le righe di commento. I commenti sono identificati da un apice all'inizio della riga.

```
<%
  'Riga1 di commento
  'Riga2 di commento
%>
```

Poiché lo script è eseguito sul server, le righe di commento non sono visibili all'utente che richiede l'esecuzione della pagina ASP dal browser.

Nel linguaggio *VBScript* non è necessario dichiarare una **variabile** prima di utilizzarla: è comunque opportuno dichiarare le variabili prima del loro uso, perché eventuali disattenzioni, con l'uso della stessa variabile in punti diversi dello script con il nome scritto in modo non corretto, potrebbero causare risultati errati.

Il programmatore può anche rendere obbligatoria la **dichiarazione esplicita delle variabili** inserendo la clausola

```
<% Option Explicit %>
```

Con questa dichiarazione, l'interprete non accetta l'uso di una variabile che non sia stata dichiarata in precedenza.

La dichiarazione di una variabile è indicata dall'istruzione **Dim**:

```
<% Dim NomeVariabile %>
```

Si possono dichiarare anche più variabili con una sola istruzione *Dim*, separando i nomi con la virgola:

```
<% Dim A, B, C %>
```

L'**assegnazione** di un valore alla variabile si rappresenta con il simbolo =. Per esempio:

```
<% A = 3 %>
```

Durante l'esecuzione dello script, l'interprete esegue automaticamente la gestione dei tipi di dati e la conversione di tipo, per cui non è obbligatoria la dichiarazione del tipo di dato.
Nello script si possono anche dichiarare valori costanti tramite l'istruzione **Const**, indicando il nome e il valore della costante:

```
<% Const Lingua = "Inglese"
   PiGreco = 3.14
   DataIniziale = #12-20-2005#
%>
```

I valori stringa sono delimitati da doppi apici, le date e le ore sono delimitate da una coppia di simboli **#**.

Gli **operatori** che si possono utilizzare nelle espressioni sono:
+, **-**, *****, **/** per le operazioni elementari
**** per la divisione intera
Mod per il resto della divisione intera
& per la concatenazione di stringhe (si può usare anche l'operatore +)
=, **<>**, **<**, **>**, **<=**, **>=** per i confronti
And, **Or**, **Not**, **Xor** per le operazioni logiche.

Le variabili di tipo **array** sono dichiarate da un'istruzione *Dim* che specifica il numero delle componenti dell'array:

```
<% Dim A(5) %>
```

Questa dichiarazione crea un array di 6 elementi, perché la numerazione inizia da 0.
Per assegnare i valori alle singole componenti dell'array, si usa l'indice racchiuso tra parentesi tonde dopo il nome dell'array:

```
<% A(0) = 345
   A(1) = 702
   A(2) = 1234
   ....
%>
```

La **struttura condizionale** si rappresenta con il seguente schema generale:

```
<%
     If (condizione) Then
        'istruzioni 1
     Else
        'istruzioni 2
     End If
%>
```

La **struttura di selezione multipla** si rappresenta con il seguente schema generale:

```
<%
   Select Case Selettore
   Case valore1, valore2, valore3
        'istruzioni 1
   Case valore4, valore5
        'istruzioni 2
   Case Else
        'istruzioni 3
   End Select
%>
```

Nel linguaggio *VBScript* si possono usare diverse strutture per la ripetizione.
La **ripetizione enumerativa** si rappresenta con la struttura **For…Next**:

```
<%
   Dim i
   For i = ValoreIniziale to ValoreFinale
        'istruzioni
   Next
%>
```

Costruire la tavola pitagorica.
Lo script costruisce una tabella di 10 righe e 10 colonne; ciascuna cella contiene il prodotto del numero di riga per il numero di colonna. La procedura usa due cicli *For* annidati.
L'esempio mostra anche come sia possibile costruire una pagina ASP integrando nello stesso testo codice *VBScript* e tag HTML. Il codice è separato dal testo e dai tag HTML attraverso le coppie di delimitatori <% … %>.

PAGINA ASP (*Tavola.asp*)

```
<%@ LANGUAGE = VBScript %>
<HTML>
<BODY>
<CENTER>
<H2>Tavola pitagorica</H2>
<TABLE BORDER=1>
<%
   Dim i, j
   For i = 1 to 10
   %>
   <TR>
   <% For j = 1 to 10 %>
          <TD ALIGN=center WIDTH="10%"> <% = i*j %> </TD>
   <% Next %>
   </TR>
   <% Next
%>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

La **ripetizione con controllo della condizione** si rappresenta con la struttura **While…Wend**:

```
<%
   While(condizione)
        'istruzioni
   Wend
%>
```

oppure con la struttura **Do While...Loop**:

```
<%
   Do While(condizione)
        'istruzioni
   Loop
%>
```

Le pagine ASP utilizzano i concetti di base della **programmazione a oggetti**, offrendo al programmatore la possibilità di definire le classi e gli oggetti.
A ogni oggetto sono associati **metodi** (procedure o funzioni) e **proprietà** (attributi o caratteristiche dell'oggetto).
A un livello elementare di implementazione di pagine ASP, si utilizzano gli **oggetti predefiniti** che risolvono le operazioni di uso più comune nelle applicazioni per il Web.
La sintassi generale per la chiamata di un metodo è la seguente:

Oggetto.Metodo *parametri*

La sintassi generale per l'utilizzo delle proprietà è la seguente:

Oggetto.Proprietà

Per alcuni oggetti è possibile visualizzare e anche impostare il valore di una proprietà.

I più importanti oggetti ASP predefiniti sono:

• Oggetto Request

per utilizzare le informazioni provenienti da una richiesta contenuta in un indirizzo URL (in formato HTTP), oppure inviata dal browser tramite un form HTML (metodi *POST* o *GET*).
Per esempio, l'istruzione:

```
<% NomeStudente = Request.Form("studente")%>
```

assegna alla variabile *NomeStudente* il valore proveniente da una casella di testo (o da un altro oggetto grafico), avente la proprietà *NAME* uguale a *studente*, contenuta nel form di una pagina HTML.

• Oggetto Response

per inviare dati che vengono visualizzati dal browser dell'utente.
Per esempio:

```
<% Response.Write "Hello World" %>
<% Response.Write "<TABLE BORDER = 1>" %>
```

Entrambe le istruzioni inviano stringhe di caratteri che devono essere interpretate dal browser dell'utente: la prima istruzione visualizza un messaggio, la seconda contiene codice HTML per creare una tabella.

• Oggetto Server

per utilizzare i metodi e le proprietà sul server.
Il metodo più frequentemente utilizzato è *Server.CreateObject*, per creare un oggetto tra quelli contenuti nelle librerie software, dette **componenti**, cioè moduli software specializzati per svolgere un insieme di operazioni e riutilizzabili in diverse applicazioni.

3. L'interazione con l'utente tramite i form HTML

Una delle caratteristiche più importanti di tutti i linguaggi di scripting è rappresentata dalla possibilità di interagire con l'utente che utilizza una pagina Web. Con le pagine ASP l'interazione avviene nel momento in cui l'utente invia una richiesta al server Web: quest'ultimo genera come risposta una pagina Web creata dinamicamente in quel momento.
L'interazione con l'utente prevede anche la possibilità di inviare alcuni dati come **parametri** della chiamata alla pagina ASP.
Il passaggio di parametri a una pagina ASP viene gestito attraverso i **form**, cioè i moduli del linguaggio HTML, che permettono la costruzione di un'interfaccia grafica, formata da caselle di testo e da pulsanti. Tramite questa interfaccia, l'utente può inserire i valori e inviarli al server Web come parametri dello script.

Creare un modulo per raccogliere i dati di un utente che desidera iscriversi a un servizio Web (cognome, nome, email).

Il modulo è formato da tre caselle di testo e da due pulsanti di comando e viene rappresentato in HTML con il seguente codice.

PAGINA HTML (*Richiesta.htm*)

```
<HTML>
<HEAD></HEAD>
<BODY>
<H1>Iscrizione al servizio Web</H1>
<FORM ACTION="/Scripts/Iscrizione.asp" METHOD="post">
Cognome:  <INPUT TYPE="text" NAME="cognome"><BR>
Nome:  <INPUT TYPE="text" NAME="nome"><BR>
E-mail:  <INPUT TYPE="text" NAME="email"><BR>
<P>
<INPUT TYPE="submit" VALUE="Invia" NAME="b1">
<INPUT TYPE="reset" VALUE="Annulla" NAME="b2"></P>
</FORM>
</BODY>
</HTML>
```

Quando si fa clic sul pulsante con l'etichetta *Invia*, il browser richiama lo script *Iscrizione.asp* indicato nell'intestazione del modulo come valore dell'attributo **ACTION**. Il file *asp* si trova nella sottodirectory *Scripts* del Web server.
Il browser aggiunge automaticamente alla richiesta tutti i campi presenti nel modulo. Per ogni campo crea un parametro, avente come nome il nome del campo (indicato dall'attributo **NAME** del tag <*INPUT*>) e come valore ciò che l'utente ha inserito nella casella di testo.
In questo modo viene realizzato il passaggio di parametri a uno script ASP tramite l'utilizzo dei *form* HTML.
L'attributo **METHOD** del tag <*FORM*> ha il compito di indicare al browser quale metodo deve utilizzare per inviare i campi del modulo al server Web.

Le modalità disponibili sono *GET* e *POST*:

- nella modalità **GET** i parametri vengono codificati automaticamente dal browser all'interno dell'indirizzo e vengono visualizzati insieme all'URL della pagina Web. Questa modalità non è adatta in tutte le situazioni: per esempio non è desiderabile che una password sia visualizzata insieme all'indirizzo. In certi casi, inoltre, non è tecnicamente possibile utilizzare la modalità *GET* viste le limitazioni sulla lunghezza degli indirizzi: si pensi a un modulo che contiene un'area di testo (*TEXTAREA*) con un numero notevole di caratteri.

- la modalità **POST** viene utilizzata per spedire grandi quantità di dati in modo che non siano visibili all'utente.

In risposta all'inserimento dei dati, il server, eseguendo la pagina ASP *Iscrizione.asp*, crea dinamicamente una pagina Web con il riassunto dei dati inseriti e chiede conferma all'utente.

PAGINA ASP (*Iscrizione.asp*)

```
<%@ LANGUAGE = VBScript %>
<%  Option Explicit %>
<%
   Dim Stringa1, Stringa2, Stringa3
   Stringa1 = Request.Form("cognome")
   Stringa2 = Request.Form("nome")
   Stringa3 = Request.Form("email")
%>
<HTML>
<BODY>
<H3>I dati che hai inserito sono </H3><HR>
Cognome : <% Response.Write Stringa1 %><BR>
Nome : <% Response.Write Stringa2 %><BR>
E-mail : <% Response.Write Stringa3 %><BR>
<HR>
<FORM ACTION="Registra.asp" METHOD="post">
<INPUT TYPE="submit" VALUE="Conferma">
</FORM>
</BODY>
</HTML>
```

La conferma provoca la chiamata di un'altra pagina ASP (*Registra.asp*) per la registrazione dei dati in un archivio. Le modalità per l'inserimento dei dati nell'archivio saranno illustrate nei prossimi paragrafi.
L'insieme **Form** dell'oggetto **Request** conserva i nomi e i valori degli elementi inseriti nei moduli HTML e inviati alla pagina ASP tramite il metodo POST.
Il metodo **Write** dell'oggetto **Response** invia al browser dell'utente i valori delle variabili specificate.

Il passaggio di parametri tramite l'indirizzo URL

Un modo diverso per passare i parametri a una pagina ASP consiste nell'**aggiungere i parametri all'indirizzo URL della pagina Web** che si sta richiamando.
L'indirizzo assume la seguente struttura generale:

pagina.asp?nome=valore

dove
nome indica il nome dal parametro che viene passato a *pagina.asp*,
valore indica il valore del parametro.

Il carattere ? separa il nome dello script dai parametri.
Gli eventuali spazi contenuti nei nomi dei parametri, o nei valori stringa assegnati, sono convertiti automaticamente in *%20*, che indica il valore esadecimale del carattere *spazio* (valore ASCII decimale = 32).
Quando viene richiesta una pagina ASP con passaggio di parametri, lo script costruisce una variabile che fa riferimento al parametro ricevuto. Il nome e il valore della variabile corrispondono al nome e al valore del parametro.

Un esempio di passaggio di parametri è il seguente:

cerca.asp?id=19

Quando il server Web riceve la precedente richiesta, inserisce automaticamente nell'insieme **QueryString** dell'oggetto **Request** la variabile *id* a cui assegna il valore 19.
È possibile passare più di un parametro utilizzando la codifica dell'indirizzo: in questo caso si devono separare i parametri, cioè le coppie *nome=valore*, tramite il simbolo **&**.
Per esempio si registri, nella sottodirectory *Scripts* del Web server, la seguente pagina ASP con il nome *Identita.asp*:

PAGINA ASP (*Identita.asp*)

```
<%@ LANGUAGE = VBScript %>
<%  Option Explicit %>
<%
   Dim Stringa1, Stringa2
   Stringa1 = Request.QueryString("Cognome mio")
   Stringa2 = Request.QueryString("Nome")
%>
<HTML>
<BODY>
<H3>Identificazione </H3><HR>
<% Response.Write Stringa1 + " " + Stringa2 %>
</BODY>
</HTML>
```

Scrivendo poi il seguente URL nella casella dell'indirizzo del browser

```
http://localhost/Scripts/identita.asp?Cognome mio=Rossi&nome=Marco
Antonio
```

poiché il nome della prima variabile e il secondo valore contengono spazi bianchi, l'indirizzo viene automaticamente convertito in

```
http://localhost/Scripts/
identita.asp?Cognome%20mio=Rossi&nome=Marco%20Antonio
```

In risposta alla richiesta, la pagina ASP invia al browser una pagina Web creata dinamicamente, contenente il cognome e il nome forniti.

4. L'accesso ai database con la tecnologia ADO

I componenti **ADO** (*ActiveX Data Objects*) forniscono gli oggetti e i metodi per la connessione ai database e per l'accesso ai dati in esso contenuti.

In particolare sono disponibili gli oggetti predefiniti:

- **ADODB.Connection**, per stabilire una connessione con l'origine dei dati, cioè con il database residente sul server;
- **ADODB.Recordset**, per conservare l'insieme delle righe della tabella ottenute come risultato di un'interrogazione oppure le righe sulle quali si effettuano le operazioni di manipolazione.

Il programmatore di pagine ASP può utilizzare questi oggetti predefiniti creando prima di tutto un'**istanza** dell'oggetto e assegnando ad essa un nome, con il metodo **Server.CreateObject**:

```
Set conn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")
```

Se sul server Web a disposizione è stata configurata l'interfaccia per i dati **OLE DB**, l'apertura della connessione richiede di specificare, come parametri, le caratteristiche del database (*provider* dell'origine dati) e il nome del database.
Per esempio, l'accesso ai database di *Microsoft Access* richiede la seguente stringa di connessione (per comodità di lettura, l'assegnazione è effettuata usando due righe):

```
strconn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
strconn = strconn + Server.MapPath("nome.mdb")
```

dove *nome.mdb* deve essere sostituito dal nome fisico (o, meglio, dal *pathname* sul disco) del database di Access.
Con queste impostazioni, la connessione viene poi aperta con l'istruzione:

```
conn.Open strconn
```

utilizzando il metodo **Open** dell'oggetto **Connection**.
I dati contenuti in una tabella sono resi disponibili con il metodo **Open** dell'oggetto **Recordset**:

```
rs.Open nome, conn
```

dove *nome* indica la tabella del database e *conn* il nome della connessione.

Leggere i record contenuti in una tabella del database, della quale viene fornito il nome da tastiera.

L'applicazione è formata da due file:
- la pagina HTML che chiede all'utente il nome della tabella da visualizzare (*Legget.htm*); la pagina è registrata nella directory principale del server Web;
- la pagina ASP che accede alla tabella del database e presenta sul video i valori contenuti nei campi di tutti i record (*Legget.asp*); lo script è registrato nella sottodirectory *Scripts* del server Web.

Il database di Access contenente la tabella si chiama *db2.mdb* ed è registrato nella stessa sottodirectory *Scripts*.

PAGINA HTML *(Legget.htm)*

```
<HTML>
<HEAD>
<TITLE>Visualizzazione tabella</TITLE>
</HEAD>
<BODY>
<H3>Richiesta nome tabella </H3>
<FORM METHOD="post" NAME="form1" ACTION="/Scripts/Legget.asp">
<P><B>Nome della tabella</B><BR>
<INPUT TYPE="text" SIZE="40" NAME="tabella"><BR>
<P><INPUT TYPE="submit" VALUE="Invia" NAME="b1">
<INPUT TYPE="reset" VALUE="Annulla" NAME="b2"></P>
</FORM>
</BODY>
</HTML>
```

PAGINA ASP *(Legget.asp)*

```
<%@ LANGUAGE = VBScript %>
<%  Option Explicit %>
<%
'dichiarazione delle variabili
   Dim conn
   Dim rs
   Dim strconn
   Dim nome
   Dim i
   nome = Request.Form("tabella")

'stringa di connessione al database
strconn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
strconn = strconn + Server.MapPath("db2.mdb")
'istanze degli oggetti Connection e Recordset
Set conn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")
'apre la connessione
conn.Open strconn
'apre il recordset
rs.Open nome, conn
%>

<HTML>
<HEAD><TITLE>Visualizzazione dei record di una tabella</TITLE></
HEAD>
<BODY>
<H3>Visualizzazione tabella</H3>
<P>
<TABLE BORDER=1>
<TR>
<%
    'nomi dei campi come intestazione delle colonne
    For i = 0 to  rs.Fields.Count - 1 %>
      <TD><B><% =rs.Fields(i).Name %></B></TD>
    <%  Next  %>
</TR>

<%
'legge le righe della tabella
    rs.MoveFirst
    Do While Not rs.EOF %>
      <TR>
      <%   For i = 0 to  rs.Fields.Count - 1 %>
           <TD><% =rs.Fields(i).Value %></TD>
      <%   Next  %>
      </TR>
      <%
      rs.MoveNext
    Loop
rs.Close
conn.Close
Set rs = nothing
Set conn = nothing
%>
```

```
</TABLE>
<BR>
</BODY>
</HTML>
```

Il nome della tabella è acquisito nello script ASP dall'insieme **Form** dell'oggetto **Request** e assegnato alla variabile *nome*:

```
nome = Request.Form("tabella")
```

essendo *tabella* il nome della casella di testo del form HTML dove l'utente ha scritto il nome della tabella richiesta.
Nella parte di script per la lettura dei record della tabella:

- *rs.Fields.Count* indica il numero di campi nel record dell'oggetto *Recordset*;
- *rs.Fields(i).Name* indica il nome di ciascun campo;
- *rs.Fields(i).Value* indica il valore di ciascun campo nel record corrente;
- *rs.MoveFirst* posiziona sul primo record del *Recordset*;
- *rs.MoveNext* posiziona sul record successivo;
- *rs.EOF* è una funzione booleana che assume il valore *true* quando viene raggiunta la fine del *Recordset*.

Questo esempio ha un carattere generale: per usare la pagina ASP con un altro database, basta semplicemente sostituire *db2.mdb* con il nome del database di Access nella definizione della stringa di connessione al database.

Esportazione dei dati da Access in ASP

La procedura più semplice per creare in modo veloce pagine ASP per l'accesso ai dati di un database è costituita dall'**esportazione** da Access.
È possibile creare file ASP da tabelle, query e maschere. I dati, nel codice HTML generato dal server, vengono visualizzati sotto forma di tabella nel browser.

Creare una pagina ASP per visualizzare i compositori di una nazione fornita come parametro.

Creiamo dapprima una query parametrica *RicercaPerNazione*, che permette di ottenere l'elenco dei compositori di una Nazione fornita da tastiera. Con la query selezionata, da menu **File**, si deve scegliere **Esporta**, indicando come tipo di file **Microsoft Active Server Pages (*.asp)**. Poiché la query è parametrica, Access simula la finestra di dialogo **Immettere valore parametro** creando una pagina HTML (*RicercaPerNazione.html*) aggiuntiva per i parametri: essa contiene un modulo HTML con una casella di testo per l'immissione dei valori dei parametri e un pulsante per l'esecuzione della query.
Si osservi che la pagina ASP e la pagina HTML, generate da Access in modo automatico, devono essere salvate in una cartella di *\inetpub\wwwroot* insieme al database.
L'esecuzione di una pagina ASP non può essere avviata direttamente con un doppio clic sopra l'icona del file *RicercaPerNazione.asp*: nella casella *Indirizzo* del browser occorre scrivere il nome della pagina HTML che richiama l'esecuzione della pagina ASP:

```
http://localhost/Musicisti/RicercaPerNazione.html
```

Si provi ad aprire il file *RicercaPerNazione.asp*, generata da Access, utilizzando un programma di editing di testi (per esempio *Blocco Note*) e si osservi il codice in esso contenuto. Si può notare che il codice generato utilizza gli stessi oggetti e le stesse modalità operative illustrate nel paragrafo precedente.

5. Esecuzione di un comando SQL

Le operazioni di manipolazione e, soprattutto, le interrogazioni sui database di un server Web, tramite pagine ASP, diventano ancora più semplici ed efficienti utilizzando i **comandi SQL** all'interno del codice degli script.
I comandi SQL possono essere *Insert*, *Update*, *Delete* per le operazioni di manipolazione e *Select* per le interrogazioni.

Leggere i record contenuti in una tabella del database, della quale viene fornito il nome da tastiera.

Questa applicazione è del tutto simile all'esempio presentato nel paragrafo precedente: la differenza consiste nell'uso del comando *Select* del linguaggio SQL e del metodo **Execute** dell'oggetto *Connection* per ritrovare i dati.

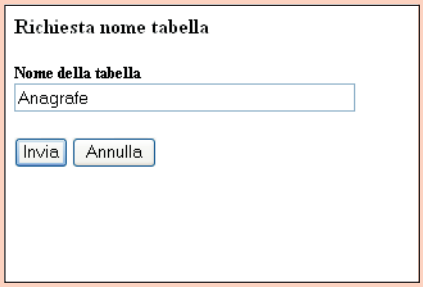
L'applicazione è formata da due file:

- la pagina HTML che chiede all'utente il nome della tabella da visualizzare (*Legget2.htm*), registrata nella directory principale del server Web;
- la pagina ASP che accede alla tabella del database e presenta sul video i valori contenuti nei campi di tutti i record (*LeggetSql.asp*); essa è registrata nella sottodirectory *Scripts*.

Questa sottodirectory contiene anche il database *db2.mdb* creato con Access.
Per usare la pagina ASP con un altro database, basta solo sostituire *db2.com* con il nome del database nell'istruzione che definisce la stringa di connessione.

PAGINA HTML *(Legget2.htm)*

```
<HTML>
<HEAD>
<TITLE>Visualizzazione tabella</TITLE>
</HEAD>
<BODY>
<H3 >Richiesta nome tabella </H3>
<FORM METHOD="post" NAME="form1" ACTION="/Scripts/LeggetSql.asp">
<P><B>Nome della tabella</B><BR>
<INPUT TYPE="text" SIZE="40" NAME="tabella"><BR>
<P><INPUT TYPE="submit" VALUE="Invia" NAME="b1">
<INPUT TYPE="reset" VALUE="Annulla" NAME="b2"></P>
</FORM>
</BODY>
</HTML>
```



Legget2.htm

LeggetSql.asp

PAGINA ASP *(LeggetSql.asp)*

```
<%@ LANGUAGE = VBScript %>
<%  Option Explicit %>
<%
'dichiarazione delle variabili
   Dim conn
   Dim rs
   Dim strconn
   Dim strSQL
   Dim i

'stringa di connessione al database
strconn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
strconn = strconn + Server.MapPath("db2.mdb")
'istanze degli oggetti Connection e Recordset
Set conn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")
conn.Open strconn
'comando SQL
strSQL = "SELECT * FROM " & Request.Form("tabella")
'oggetto Recordset
Set rs = conn.Execute(strSQL)
%>

<HTML>
<HEAD>
<TITLE>Visualizzazione dei record di una tabella</TITLE>
</HEAD>
<BODY>
<H3>Visualizzazione tabella</H3>
<P>
<TABLE BORDER=1>
<TR>
<%  For i = 0 to  rs.Fields.Count - 1 %>
      <TD><B><% =rs.Fields(i).Name %></B></TD>
<%  Next  %>
</TR>

<%
    rs.MoveFirst
    Do While Not rs.EOF %>
      <TR>
      <%   For i = 0 to  rs.Fields.Count - 1 %>
           <TD><% =rs(i) %></TD>
      <%   Next  %>
      </TR>
      <%
      rs.MoveNext
    Loop
%>
```

```
</TABLE>
<%
rs.Close
conn.Close
set rs = nothing
set conn = nothing
%>
</BODY>
</HTML>
```

Si osservi l'uso della notazione più compatta per visualizzare i valori contenuti nelle righe della tabella:

```
<% =rs(i) %>
```

che è equivalente a quella utilizzata nell'esempio del paragrafo precedente:

```
<% =rs.Fields(i).Value %>
```

Si ricordi inoltre, come già visto nell'Unità di apprendimento 8, che nella costruzione della stringa SQL contenente il comando da eseguire, si deve fare particolare attenzione alla sintassi con l'uso delle virgolette e degli spazi: se le variabili sono di tipo numerico, possono essere concatenate senza delimitazione; per le variabili di tipo stringa occorre usare invece l'apice come delimitatore.

Per esempio, per visualizzare il cognome e nome dei dipendenti che abitano in una provincia prefissata e che hanno uno stipendio superiore a una cifra minima prefissata, supponendo che i dati prefissati (*provincia* e *minimo*) vengano passati alla pagina ASP tramite un form HTML, si deve scrivere la seguente stringa per il comando SQL:

```
strSQL = "Select Cognome, Nome From Dipendenti "
strSQL = strSQL & "Where Provincia = '"
strSQL = strSQL & Request.Form("provincia") &  "'"
strSQL = strSQL & " And Stipendio > " & Request.Form("minimo")
```

La stringa è stata costruita in tre passaggi per una maggiore chiarezza di lettura.
Si noti l'uso degli apici prima e dopo la concatenazione della variabile di tipo stringa.

Uso dei comandi Insert e Update del linguaggio SQL all'interno delle pagine ASP

Gli esempi seguenti mostrano come si possano realizzare le operazioni di inserimento e di aggiornamento dei record di una tabella, utilizzando i comandi di manipolazione *Insert* e *Update* del linguaggio SQL all'interno del codice ASP.

Anche in questi esempi si utilizza il metodo **Execute** dell'oggetto *Connection* per avviare l'esecuzione del comando SQL.

Gli esempi fanno riferimento a una tabella *Anagrafe* in un database *Rubrica* di Access, avente tre campi: *ID* (di tipo *Contatore* e chiave primaria), *Cognome*, *Nome* e *Telefono*, per formare una semplice rubrica telefonica.

Creare una pagina Web per inserire nuovi nomi nel database di una rubrica telefonica, utilizzando un comando SQL.

Anche in questo caso l'applicazione è formata da due file:

- la pagina Web in formato HTML per l'inserimento dei dati (*Ingresso.htm*), registrata nella directory principale del server Web;
- una pagina ASP per la registrazione dei dati nel database (*AddSql.asp*), inserita nella sottodirectory *Scripts*.

PAGINA HTML *(Ingresso.htm)*

```
<HTML>
<HEAD>
<TITLE>Inserimento di record con SQL</TITLE>
</HEAD>
<BODY>
<CENTER><H3>Inserimento nuovi record con SQL</H3></CENTER>
<FORM METHOD="post" NAME="form1" ACTION="/Scripts/AddSql.asp">
<P><B>Cognome</B><BR>
<INPUT TYPE="text" SIZE="40" NAME="Cognome"><BR>
<B>Nome</B><BR>
<INPUT TYPE="text" SIZE="40" NAME="Nome"><BR>
<B>Telefono</B><BR>
<INPUT TYPE="text" SIZE="40" NAME="Telefono"><BR>
<P><INPUT TYPE="submit" VALUE="Invia" NAME="b1">
<INPUT TYPE="reset" VALUE="Annulla" NAME="b2"></P>
</FORM>
</BODY>
</HTML>
```

La pagina Web è del tutto simile alla precedente, è stata riscritta per comodità di lettura: l'unica differenza consiste nel nome dello script ASP che viene attivato quando l'utente fa un clic sul pulsante *Invia* (in questo caso *AddSql.asp*).
Il testo della pagina ASP è riportato sotto.

PAGINA ASP *(AddSql.asp)*

```
<%@ LANGUAGE = VBScript %>
<%  Option Explicit %>
<%
'Dichiarazione delle variabili
   Dim conn
   Dim strconn
   Dim strSQL

'stringa di connessione al database
strconn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
strconn = strconn + Server.MapPath("Rubrica.mdb")
'istanza dell'oggetto Connection
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open strconn
```

```
'costruzione del comando SQL con gli input del form HTML
strSQL = "INSERT INTO Anagrafe(Cognome, Nome, Telefono)"
strSQL = strSQL & " VALUES ("
strSQL = strSQL & "'" & Request.Form("Cognome") & "',"
strSQL = strSQL & "'" & Request.Form("Nome") & "',"
strSQL = strSQL & "'" & Request.Form("Telefono") & "'"
strSQL = strSQL & ")"

'metodo Execute dell'oggetto Connection per inserire il record
conn.Execute(strSQL)
conn.Close
set conn = nothing
%>

<HTML>
<HEAD>
<TITLE>Esempio di inserimento di record con SQL</TITLE>
</HEAD>
<BODY>
<H2><% Response.Write "Il record è stato registrato" %></H2>
<P><A HREF="/Ingresso.htm">Torna alla pagina di inserimento</A>
</BODY>
</HTML>
```

In questo caso non viene usato alcun oggetto *Recordset*, in quanto la registrazione di un nuovo record nel database si realizza applicando all'oggetto **Connection** il metodo **Execute**, il quale lancia l'esecuzione del comando SQL con *INSERT INTO*.
Si osservi che la sintassi generale del comando INSERT INTO del linguaggio SQL è la seguente:

```
INSERT INTO NomeTabella (Campo1, Campo2, ... , Campon)
VALUES ('valore1','valore2', ... , 'valoren')
```

Utilizzando i valori provenienti dal modulo HTML, con i dati inseriti dall'utente, si deve costruire la frase SQL come indicato nel codice precedente:

```
'costruzione del comando SQL con gli input del form HTML
strSQL = "INSERT INTO Anagrafe(Cognome, Nome, Telefono)"
strSQL = strSQL & " VALUES ("
strSQL = strSQL & "'" & Request.Form("Cognome") & "',"
strSQL = strSQL & "'" & Request.Form("Nome") & "',"
strSQL = strSQL & "'" & Request.Form("Telefono") & "'"
strSQL = strSQL & ")"
```

Aggiornare un record della rubrica telefonica del quale viene fornito il valore del campo chiave.

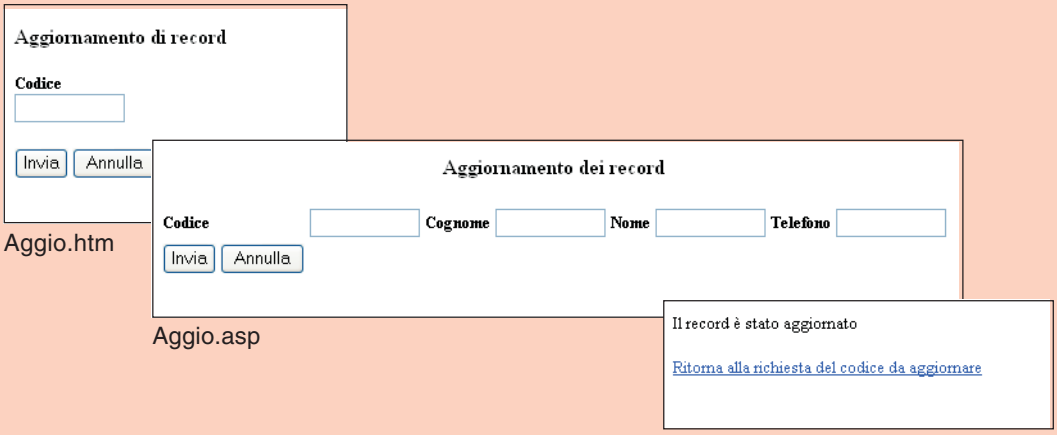
L'applicazione è realizzata mediante tre file:

- la pagina HTML che chiede all'utente il codice del record da modificare (*Aggio.htm*), registrata nella directory principale del server Web;
- la pagina ASP che accede alla tabella del database e presenta sul video i valori contenuti nei campi del record, chiedendo all'utente i nuovi valori (*Aggio.asp*); essa è registrata nella sottodirectory *Scripts*;
- la pagina ASP che aggiorna il record nella tabella con i nuovi valori acquisiti (*Registra.asp*), registrata nella sottodirectory *Scripts*.

Questa sottodirectory contiene anche il file *Rubrica.mdb* del database di Access.

PAGINA HTML *(Aggio.htm)*

```
<HTML>
<HEAD>
<TITLE>Aggiornamento di record con SQL</TITLE>
</HEAD>
<BODY>
<H3>Aggiornamento di record</H3>
<FORM METHOD="post" NAME="form1" ACTION="/Scripts/Aggio.asp">
<P><B>Codice</B><BR>
<INPUT TYPE="text" SIZE="10" NAME="Codice"><BR>
<P><INPUT TYPE="submit" VALUE="Invia" NAME="b1">
<INPUT TYPE="reset" VALUE="Annulla" NAME="b2"></P>
</FORM>
</BODY>
</HTML>
```



PAGINA ASP *(Aggio.asp)*

```
<%@ LANGUAGE = VBScript %>
<%  Option Explicit %>
<%
'Dichiarazione delle variabili
   Dim conn
   Dim rs
   Dim strconn
   Dim strSQL

'stringa di connessione al database
strconn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
strconn = strconn + Server.MapPath("Rubrica.mdb")
'istanze degli oggetti Connection e Recordset
Set conn = Server.CreateObject("ADODB.Connection")
```

```
Set rs = Server.CreateObject("ADODB.Recordset")
conn.Open strconn
'comando SQL
strSQL = "SELECT * FROM Anagrafe WHERE ID = " &
Request.Form("codice")
'oggetto Recordset
Set rs = conn.Execute(strSQL)
%>

<HTML>
<HEAD>
<TITLE>Aggiorna tabella</TITLE>
</HEAD>
<BODY>
<CENTER><H3>Aggiornamento dei record</H3></CENTER>
<FORM METHOD="post" NAME="form1" ACTION="/Scripts/Registra.asp">
<TABLE>
<TR>
<TD><B>Codice</B></TD>
<TD><INPUT TYPE="text" SIZE="10" NAME="ID"
VALUE="<% =rs("ID")%>"></TD>
<TD><B>Cognome</B></TD>
<TD><INPUT TYPE="text" SIZE="10" NAME="Cognome"
VALUE="<% =rs("Cognome")%>"></TD>
<TD><B>Nome</B></TD>
<TD><INPUT TYPE="text" SIZE="10" NAME="Nome"
VALUE="<% =rs("Nome")%>"></TD>
<TD><B>Telefono</B></TD>
<TD><INPUT TYPE="text" SIZE="10" NAME="Telefono"
VALUE="<% =rs("Telefono")%>"></TD>
</TR>
<TR>
<TD><INPUT TYPE="submit" VALUE="Invia" NAME="b1">
<INPUT TYPE="reset" VALUE="Annulla" NAME="b2"></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>

<%
rs.Close
conn.Close
'rilascia gli oggetti
Set rs = nothing
Set conn = nothing
%>
```

PAGINA ASP *(Registra.asp)*

```
<%@ LANGUAGE = VBScript %>
<%  Option Explicit %>
<%
'Dichiarazione delle variabili
   Dim conn
   Dim strconn
   Dim strSQL

'stringa di connessione al database
strconn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
strconn = strconn + Server.MapPath("Rubrica.mdb")
'istanza dell'oggetto Connection
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open strconn

'comando SQL
strSQL="UPDATE Anagrafe "
strSQL=strSQL & "SET Anagrafe.Cognome='" & Request.Form("Cognome")
& "', "
strSQL=strSQL & "Anagrafe.Nome='" & Request.Form("Nome") & "', "
strSQL=strSQL & "Anagrafe.Telefono='" & Request.Form("Telefono") &
"' "
strSQL=strSQL & "WHERE Anagrafe.ID=" & Request.Form("ID")
conn.Execute(strSQL)

conn.Close
set conn = nothing
%>

<HTML>
<HEAD>
<TITLE>Conferma dell'aggiornamento</TITLE>
</HEAD>
<BODY>
<% Response.Write "Il record è stato aggiornato" %>
<P><A HREF = "/Aggio.htm">Ritorna alla richiesta del codice da
aggiornare</A>
<P>
</BODY>
</HTML>
```

La pagina *Aggio.asp* utilizza una query SQL con il comando *Select* per costruire un *Recordset,* al quale poi lo script accede per ritrovare i dati. Nella seconda pagina *Registra.asp*, l'aggiornamento del record viene realizzato tramite l'esecuzione del comando *Update .... Set*, che nel linguaggio SQL indica l'operazione di modifica del contenuto dei campi di un record.

ESERCIZI

1 Quali delle seguenti affermazioni riferite alle pagine ASP sono vere (V) e quali false (F)?

a) Per applicazione ASP si intende l'insieme di tutte le pagine ASP del server Web V F
b) L'oggetto Response serve a inviare dati che vengono visualizzati dal browser dell'utente V F
c) Con il metodo Response.Write si può inviare al browser anche codice HTML V F
d) L'oggetto Request gestisce solo le informazioni provenienti da un form HTML V F

2 Quale delle seguenti istruzioni è scritta in modo corretto per assegnare alla variabile *Nome* il valore proveniente da una casella di testo avente la proprietà *Name* uguale a *Testo1*, contenuta nel form di una pagina HTML?

a) <% Testo1 = Request.Form("Nome") %>
b) <% Nome = Form("Testo1") %>
c) <% Request.Nome = Form("Testo1") %>
d) <% Nome = Request.Form("Testo1") %>

3 Qual è la codifica corretta dell'URL per passare due parametri (codice e provincia) allo script cerca.asp?

a) cerca.asp&codice=19?provincia=PN
b) cerca.asp?codice=19;provincia=PN
c) cerca.asp?codice=19&provincia=PN
d) cerca.asp!codice=19&provincia=PN

4 Quale delle seguenti istruzioni crea un'istanza dell'oggetto ADODB.Connection?

a) Set conn = Server.CreateObject("ADODB.Connection")
b) Set Server.conn = CreateObject("ADODB.Connection")
c) Set conn = CreateObject.Server("ADODB.Connection")
d) Set Server.conn = Object("ADODB.Connection")

5 Completa le frasi seguenti utilizzando una tra le parole indicate alla fine della domanda

a) L'oggetto per stabilire una connessione al database è .....
b) L'istruzione per aprire una connessione *conn* è .....
c) L'oggetto per gestire un Recordset è .....
d) L'istruzione per aprire un Recordset *rs* è .....

Request.Form, Response.Write, Server.CreateObject, ADODB.Connection, ADODB.Recordset, conn.Open, rs.Open, conn.Close, rs.Close

6 Quale delle seguenti istruzioni inserisce in un oggetto Recordset *rs* le righe ottenute come risultato di un'interrogazione SQL contenuta nella stringa *strSQL*?

a) Set rs = conn.Execute(strSQL)
b) Set strSQL = conn.rs
c) Set rs = conn.strSQL
d) Set rs.conn = Execute strSQL