A. Veneziani - Classe V Abacus -

Elementi di elaborazione Web con supporto ai DB basati su ASP e Access

Prologo

Parlando di tecnologie Web legate a Microsoft è naturale integrare le funzioni di ASP[1] con le altre tecnologie e prodotti della stessa Microsoft, anche se si deve tener presente che ASP è integrabile con moltissimi DBMS[2] presenti sul mercato. Questo grazie ad una architettura modulare che utilizza diversi cosiddetti driver o provider per connettersi a diversi DBMS. La libreria ADO (ActiveX Data Objects) è quanto di più efficace offre la "vecchia" tecnologia COM / ActiveX di Microsoft; si tratta di un componente atto a effettuare connessioni tra programmi desktop o applicazioni Web e DBMS presenti in locale od in remoto.

La struttura del collegamento, come riportato dalla stessa documentazione Microsoft, si suddivide in pratica in 2 livelli, di cui uno di questi è la libreria ADO stessa, costituita come un componente ActiveX standard, che fornisce una interfaccia unificata per le connessioni ed il trattamento dei dati dei vari DBMS, ed un altro è il livello dei driver, che ovviamente si differenziano caso per caso, a seconda del DBMS utilizzato. Ovviamente sopra il livello ADO, troviamo l'applicazione Windows[3] che utilizza ADO[4], mentre sotto il livello driver troviamo il DBMS stesso a cui ci si deve opportunamente connettere ed utilizzare.

Il DB

Per il nostro progettino abbiamo utilizzato Access 2007, creando in esso due tabelle, una, ovvia che contiene i dati elementari degli utenti, in questo caso rappresentati da Login, Password ed una chiave primaria (un Id_Utente che usualmente è la chiave primaria della tabella).

A questi dati si potrebbe anche aggiungerne degli altri per rendere più completo e verosimile il contenuto della tabella Utenti stessa, ad esempio altri dettagli riguardanti l'utente quali nome, cognome ed altro, così che una volta riconosciuto all'autenticazione essi possano essere utili per diversi scopi e non ultimo per tenere traccia precisa dei riferimenti dell'utenza stessa. In pratica per la tabella Utenti avremo i campi:

Id_Utente	Numerico Chiave primaria Contatore
Login	Alfanumerico (50 caratteri)
Password	Alfanumerico (50 caratteri)

Al DB è stata poi aggiunta un altra tabella atta a rilevare quando l'utente acceda o meno al sistema, tramite registrazione di alcune informazioni nel database. Questi dati ovviamente

1 Ricordo per inciso che questa tecnologia già da diversi anni è affiancata dalla più complessa e moderna ASP.NET basata sul framework Microsoft .NET, ma che comunque ASP risulta ancora supportato ed è utilizzabile anche sulle nuove versioni di IIS e su Windows 8.

2 DBMS (DataBase Managment System) – sigla che stà per ad indicare i numerosi prodotti sw che servono a gestire e far funzionare database, quali Access, SQL Server, Oracle, MySql ed altri.

3 Si fa presente ancora che ADO è una tecnologia Microsoft e quindi presente specificamente solo sui sistemi Windows.

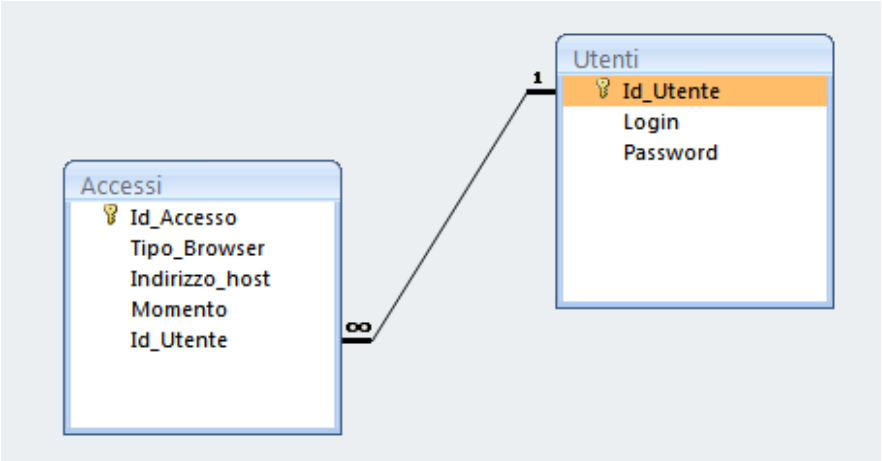
4 Nel nostro caso l'applicazione non è una classica applicazione desktop, ma una applicazione costituita da script ASP, e quindi una cosiddetta applicazione Web.

vengono inseriti in una tabella separata apposita, che abbiamo chiamato Accessi.

Accessi è stata così conformata come:

Id_Accesso	Numerico Chiave primaria Contatore
Tipo_Browser	Alfanumerico (50 caratteri)
Indirizzo_Host	Alfanumerico (50 caratteri)
Momento	Data / ora
Id_Utente	Numerico

ove Id_Utente risulta quindi chiave esterna nella relazione che lega la tabella Utenti alla tabella accessi. E' possibile infatti tramite la opportuna lettura di dati provenienti dal browser ricavare il sistema operativo che sta utilizzando l'utente, il tipo e la versione di browser, ed anche il numero di IP da cui arriva la chiamata al server. Tutte queste possono essere informazioni opportune da registrare per tracciare opportunamente i vari accessi ad una applicazione Web. L'ultimo campo conterrà l'id dell utente che ha effettuato l'accesso, ed è quindi ad esso che si riferiscono le informazioni degli altri campi.



Il database verrà poi salvato e memorizzato in un opportuno file .accdb, o all'occorrenza salvato nel vecchio formato Access[5] .mdb.

Stabilire la connessione

Nel nostro esempio di relativo alla autenticazione tramite consultazione di un apposito DB Access, la prima operazione della pagina ASP preposta alla autenticazione era quella di connettersi al DB stesso. Per fare ciò un sistema Microsoft offre varie tecniche di connessione, tra cui ODBC, e viceversa un collegamento diretto indicato spesso come DSN-less.

Il tipo di connessione che noi effettueremo tramite ADO è del tipo DSN-less, ossia senza alcuna operazione di settaggio previa sulla macchina, atta a fornire un riferimento di connessione al database.

In questo caso, siccome sono comunque necessari alcuni riferimenti ben precisi per effettuare la connessione, è necessario comunicare ad ADO questi dati, tramite una cosiddetta connection

5 Attenzione alcuni comportamenti, le connection string necessarie alla connessione ed altri dettagli non sono del tutto analoghi tra le vecchie versioni (es. 2003) e la versione 2007.

string, che ha diverse varianti, a seconda del DBMS utilizzato, ma anche del tipo di accesso richiesto sul DB e di altri parametri. La connectionstring nel nostro caso contiene il riferimento ad un driver indicato come *Provider*, che è un opportuno modulo software che si interfaccia tra ADO ed il DB permettendo di adattare l'uso della libreria a potenzialmente qualunque database di concezione recente. Le connectionstring per Access indicano poi quasi sempre la posizione del file .mdb o .accdb[6] tramite un percorso assoluto in un apposito campo indicato con *Data Source*.. Talora alcuni esempi di connectionstring propongono anche l'indicazione di un utenza, ma questo e' relativamente importante siccome sappiamo che spesso Access non è utilizzato distinguendo tra varie utenze.

La connectionstring viene inserita settando la proprietà ConnectionString dell'oggetto Connection (che in questo caso si chiama ConnObj). Nel nostro caso una tipica stringa di connessione potrà essere:

```
ConnObj.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Inetpub\wwwroot\miaroot\esempi_scuola\autenticazione_db\autenticazione.accdb;"
```

(La stringa qui sopra è associata ad un DB Access 2007).

In generale tutte le connectionstring sono costituite da una successione di campi e valori, con vari significati, separati dal carattere " ;", ossia del tipo:

```
"<campo1>=<valore1>;<campo2>=<valore2>;...."
```

Successivamente all'assegnazione della connectionstring alla proprietà omonima dell'oggetto Connection, si dà il comando di aprire la connessione ossia `ConnObj.Open`, applicando il metodo Open all'oggetto stesso.

Effettuare comandi SQL

A seguire è possibile inviare al DB qualunque comando SQL, sia senza ritorno di dati (tipicamente INSERT INTO, UPDATE, ecc.) , sia con ritorno di dati (SELECT...FROM...).

Nel nostro caso il tipico comando per individuare i vari utenti, leggerne la login e password e determinare se coincide con quella inserita dall'utente è:

```
Set RsObj = ConnObj.Execute("SELECT * FROM Utenti")
```

Come si vede si tratta solo di mettere tra le parentesi del metodo Execute l'opportuno comando SQL da eseguire. Il risultato del comando, in questo caso[7],è un oggetto recordset, che viene opportunamente istanziato oltrechè riempito di contenuti. Un recordset, come dice lo stesso nome, è un insieme di record. Ogni riga estratta dalla tabella (come noto non è detto che una SELECT estragga tutte le righe, ovviamente) diviene quindi un record del recordset. Il recordset è quindi un oggetto atto ad elaborare i dati estratti dal DB tramite una query. La metodica di elaborazione del recordset prevede la presenza di un cursore (ovviamente invisibile) che tenga il segno del punto dove è arrivata la lettura. Un opportuno

6 Si ricorda che questa nuova estensione (.accdb) è stata inserita con Access 2007.

7 Come si è già detto, altri comandi non rendono valori di ritorno, quali comandi di inserzione di un record o aggiornamento di qualche dato del DB.

metodo (MoveNext) applicato all'oggetto Recordset ricavato dai vari comandi SQL[8], permette di muovere il cursore in avanti[9], andando a puntare e quindi permettendo di prelevare un nuovo record. Individuato il record, la lettura dei suoi dati è piuttosto semplice, trattandosi di indicare il nome del recordset seguito da parentesi e poi una indicazione riportante il nome del campo da leggere. In definitiva,ad esempio:

```
If RsObj("Login") = Request.Form("log") And RsObj("Password") = Request.Form("pass") Then
```

si tratta di una forma come quella indicata qui sopra, in questo caso i dati letti vengono paragonati ai dati letti dalla form sulla pagina Web di ingresso.

Ovviamente poiché quello di lettura è un procedimento da iterare per un numero di volte non noto, è necessario prevedere appositi controlli per individuare quando la lettura del recordset deve terminare ossia quando si sono considerati tutti i record.

Questo avviene con la lettura di un apposita proprietà (EOF), con significato e logica del tutto analoga a quella usata per i file. Troviamo quindi nel nostro listato la seguente riga di codice:

```
Do While Not RsObj.EOF
```

che implementa un ciclo che poi viene chiuso poco più sotto. Si tratta del ciclo di lettura dei dati, che permette di accedere ai dati estratti tramite gli appositi comandi SQL dati precedentemente. In questo ciclo sarà sempre presente anche il metodo per far muovere opportunamente il cursore (usualmente in avanti, ossia MoveNext[10]).

All'interno del ciclo abbiamo quindi potuto accedere via via ai dati tramite l'indicazione RsObj("<nome campo>"), che rende specificamente il valore contenuto nel nome del campo. Il tipo di dato reso sarà coerente con quello letto nel DB.

Nel caso login e password coincidano entrambe con quelle lette dal DB e quindi siano riferite ad un certo utente, una tecnica usuale di autenticazione prevede di settare opportunamente una variabile di sessione di modo che l'autenticazione e l'identificazione dell'utente sia confermata anche alle pagine successive dell'ipotetico sito[11].

Nel nostro progettino abbiamo poi inserito anche una funzionalità aggiuntiva, per rendere più interessante il test, vale a dire il tracciamento degli accessi degli utenti. In questo caso si raccolgono alcune informazioni, come id dell'utente che ha acceduto, la data e ora di accesso, il tipo del sistema operativo, il tipo di browser utilizzato, l'IP da cui ha effettuato la connessione e quindi l'accesso. Ovviamente si tratta solo di un possibile esempio.

Questa funzionalità è realizzata inserendo un opportuno comando SQL, dopo il controllo della corrispondenza login / password:

```
ConnObj.Execute("INSERT INTO " & _
"Accessi(Tipo_Browser,Indirizzo_Host,Momento,Id_Utente) VALUES('" & _
Request.ServerVariables("HTTP_USER_AGENT") & "','" & _
Request.ServerVariables("REMOTE_ADDR") & "','" & _
```

8 Su stà parlando di comandi ovviamente che rendano dati, usualmente SELECT.

9 In alcune modalità operative ADO permette anche di muovere il cursore all'indietro, ma comunemente il movimento sempre previsto e consentito è in avanti.

10 Si ricorda che è generalmente applicabile ad un oggetto Recorset un altro metodo detto MoveFirst per ri-posizionarsi all'inizio del recordset, ad esempio nella necessità di effettuare una seconda scansione dello stesso.

11 Nel nostro esempio queste pagine non esistono, ma è ovvio che l'autenticazione serve appunto a permettere l'ingresso in un area protetta di un sito costituito solitamente da numerose di pagine collegate.

```
        Now() & "'," & Session("IdUtente") & ")")
```

Come si potrà notare il comando SQL in questo caso non rende alcun valore, ed infatti non viene invocato alcun oggetto recordset a ricavare dati di ritorno. Il comando in questione è del tipo INSERT INTO.... perchè nella apposita tabella Accessi del DB deve essere aggiunto un record per ogni accesso effettuato con successo. Il comando però, ovviamente, non può avere valori fissi da inserire, ma essi variano a seconda dei casi. Ecco perchè nella stringa del comando sono indicati delle parti parametriche, ossia parti variabili concatenate alla parte fissa del comando. Queste parti parametriche, in molti casi sono ricavate grazie al comando Request.ServerVariables("<nome indicatore>"), che permette di leggere alcuni dei numerosi parametri disponibili al server, dopo una richiesta del client.

Si noti anche che il comando SQL, in questo caso, indica esplicitamente una lista dei campi da aggiornare (precisamente gli ultimi 4), escludendo quindi il campo Id_Accesso che è il campo contatore. Il valore di tale campo, come previsto da questo tipo di campo progredisce autonomamente, e non va quindi imposto dal programma e dal comando SQL che aggiunga un record. Di conseguenza nella parte VALUES, i valori indicati sono 4 e non 5, come sarebbe stato se si fosse imposto (in modo erroneo) un valore anche al campo contatore. Si tenga presente questa regola generale per ogni operazione di INSERT che debba operare su tabelle dotate di campi contatore.

Codice del progettino di esempio

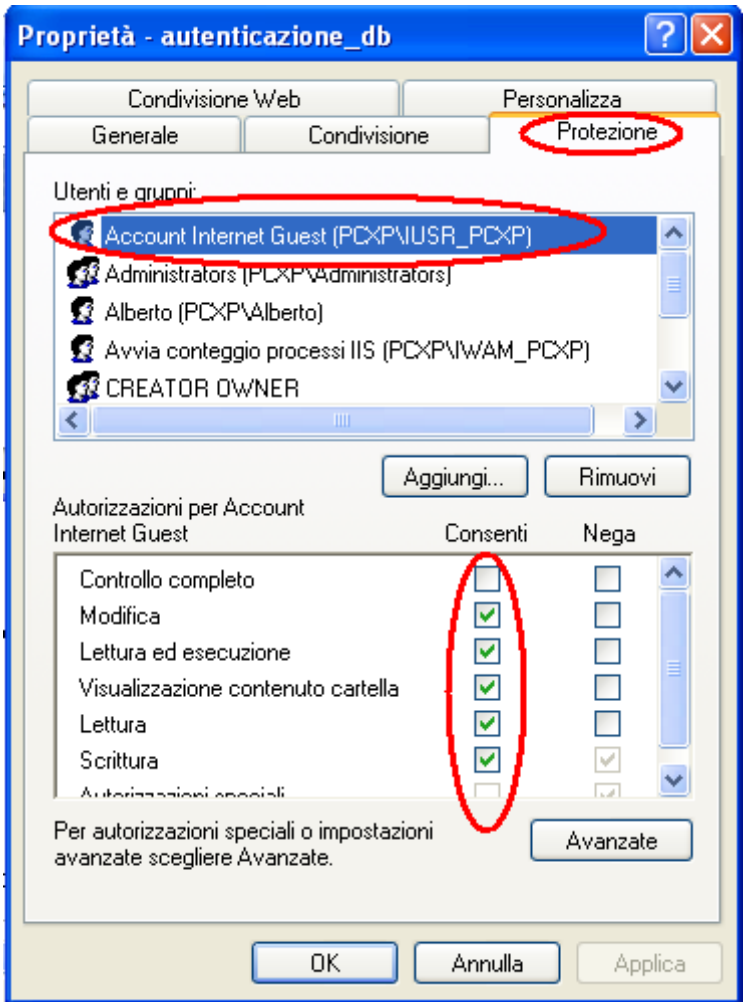
Concludiamo riportando il codice completo del progettino proposto come primo esempio di operazioni sui DB tramite ASP. Ovviamente questo è solo un esempio assai semplice di applicazione, ed operazioni ben più complesse possono essere implementate e coinvolgere i dati di molteplici tabelle:

```
<%@LANGUAGE="VBSCRIPT" CODEPAGE="65001"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Documento senza titolo</title>
</head>
<%

   ' istanzia l'oggetto  ADO per l'accesso ai DBMS
   Set ConnObj = Server.CreateObject("ADODB.Connection")
   ' In questo caso imposta una stringa adatta per DB Access versione 2007
   ConnObj.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Inetpub\wwwroot\miaroot\esempi_scuola\autenticazione_db\autenticazione.accdb;"
   ' comando di apertura del DB
   ConnObj.Open

   ' query per estrarre i dati di accesso di tutti gli utenti
   Set RsObj = ConnObj.Execute("SELECT * FROM Utenti")

   Entrato = False   ' flag che presume che l'utente non abbia autorizzazione
   Do While Not RsObj.EOF
       ' controlla se l'utente e quello attualmente considerato nella lettura dal DB
       If RsObj("Login") = Request.Form("log") And RsObj("Password") = Request.Form("pass")
Then
```

```
            ' se lo e' allora indica una autenticazione corretta...
          Response.Write "Utente autenticato !"
             Session("IdUtente") = RsObj("Id_Utente")

        'registrazione dell'avvenuto accesso con meorizzazione di alcuni dati dell'utente
              ConnObj.Execute("INSERT INTO " & _
                    "Accessi(Tipo_Browser,Indirizzo_Host,Momento,Id_Utente) VALUES('" & _
                     Request.ServerVariables("HTTP_USER_AGENT") & "','" & _
                     Request.ServerVariables("REMOTE_ADDR") & "','" & _
                     Now() & "'," & Session("IdUtente") & ")"       )

              Entrato = True     ' e' stato riconosciuto come utente registrato
          End If
          RsObj.MoveNext    ' movimento al record successivo
   Loop
   ' altrimenti non pu entrare....
   If Not Entrato  Then  Response.Write "Login o password non corrette."

%>

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

Ricordiamo che il codice sopra riportato è quello della sola pagina che effettua l'autenticazione.

Permessi di accesso in scrittura



Analogamente alla scrittura sui file, anche in questo caso non è possibile scrivere (operazioni di INSERT INTO o UPDATE o altre simili) sul DB Access, senza aver regolato opportuni permessi di accesso in scrittura per l'utente IUSR_<nome macchina>, anzi in alcuni casi tali permessi devono essere regolati sulla intera cartella dove risiede il DB, piuttosto che sul solo file. Questa problematica, in genere, non è presente usando altri database già pensati per il funzionamento centralizzato in rete quali SQL Server o MySQL, a differenza di quanto avviene con Access.

Qui a lato si riporta il settaggio di un aumento di permessi (attivazione dei permessi in scrittura) per l'utente IUSR_<nome macchina>[12] (in questo caso IUSR_PCXP).

12 L'utente IUSR_<nome_macchina> corrisponde ad un utente generico che opera nella macchina in modo controllato, tramite script ASP, o comunque interagendo col server, WWW, tramite IIS, e quindi i suoi criteri di sicurezza. Anche questo utente sottostà però alle regolazioni dei permessi definite dal sistema operativo.

Alcuni possibili errori e loro perchè

La tabella 'Utenti' è già aperta con accesso
esclusivo da un altro utente o tramite l'interfaccia
utente e non può essere manipolata a livello di programmazione.
Causa:
L'interfaccia di Access è aperta operando sul file di DB e
questo impedisce ulteriori operazioni

Impossibile usare
"C:\Inetpub\wwwroot\miaroot\esempi_scuola\autenticazione_db\autenticazione.accdb".
File già in uso.
Causa:
Analoga a sopra

Il numero dei valori nella query non corrisponde
a quello dei campi di destinazione.
Causa:
Errore che di solito si verifica nelle operazioni
che usano INSERT INTO. La tabella ha un certo numero di campi ed il comando ha tentato di inserire un numero minore o maggiore di campi.
Possibile rimedio utilizzare la forma dell'istruzione INSERT INTO Tabella(Campo1, Campo2,...) che permette di indicare solo alcuni campi sul quale effettuare l'inserimento.
Tutti i campi rimanenti vengono inseriti col loro valore di default.

Per l'operazione è necessaria una query aggiornabile.
Causa:
Mancanza di permessi di accesso in scrittura al file del DB o alla cartella
(in alcuni casi è necessario estendere a tutta la cartella i permessi in scrittura)

"C:\Inetpub\wwwroot\miaroot\esempi_scuola\autenticazione_db_\autenticazione.accdb"
non è un percorso valido.
Assicurarsi che il nome del percorso sia corretto e di essere collegati
al server in cui si trova il file.
Causa:
Il percorso o il nome del file non punta correttamente al file Access stesso