Lettura di un DataBase Remoto con ADO tramite XML

http://www.vbsimple.net/activity/act_15.htm

Richiesta di: Antonio - 16 Giugno 2001
Difficoltà: 4 / 5

Come fare per leggere il contenuto di una tabella che si trova su un sito Internet remoto?

Questo articolo implementa una soluzione sviluppata da Moreno Sirri e pubblicata sul suo sito (http://www.msvbsite.net).

Con il sempre più crescente sviluppo della tecnologia Microsoft COM (Component Object Model) aumentano i programmi che utilizzano risorse distribuite su vari server.
Le ultime versioni di ADO (ActiveX Data Objects) permettono l'apertura di un Recordset senza connessione attraverso la chiamata di un indirizzo Internet HTTP o FTP.
Naturalmente è necessario che l'indirizzo Internet richiamato restituisca in uscita un Recordset.
Nota bene: abbiamo detto Recordset, non Database. Non dovremo inserire la posizione in cui si trova l'archivio dati ma dovremo provvedere una sequenza dati in una forma comprensibile all'ADO.

Tale forma di scambio di dati è l'XML (Extended Markup Language), un linguaggio di marcatura in grado di trasferire dati di qualunque genere, nato come evoluzione dell'HTML utilizzato per disegnare le pagine Web.

Il nostro progetto si compone di due parti: la prima è scritta in ASP (Active Server Pages) e servirà come interfaccia tra il nostro programmino scritto in VB ed il database remoto.
La pagina ASP interrogherà il Database per noi e restituirà un Recordset XML. Il nome del Recordset richiesto sarà passato attraverso il metodo *GET*.
La seconda parte è un semplicissimo programma in Visual Basic che richiama la pagina ASP sul server e mostra in una griglia il contenuto del Recordset ricevuto.
Iniziamo a vedere il codice della pagina ASP:

```
 1. <%
 2.   Option Explicit
 3.   Const adLockPessimistic = 2
 4.   Const adOpenkeyset = 1
 5.   Const adPeRSistXML = 1
 6.   Const adStateOpen = 1
 7.   Const adVarChar = 200
 8.   Const adInteger = 3
 9.   Const NOMEDB = "Libri.mdb"
10.   Dim CONN
11.   Dim RS
12.   Dim XMLStream
13.
```

Alle righe 3-8 sono dichiarate alcune costanti di ADO. In Visual Basic normalmente non

 Copyright Visual Basic Simple 2002

sono ncessarie, ma poiché stiamo utilizzando ASP esse non sono presenti. Suppliamo alla loro mancanza dichiarandole manualmente.
Alla riga 9 dichiariamo la variabile **NOMEDB** che conterrà il nome del database da aprire nella cartella in cui si trova la nostra pagina ASP.

Seguono le dichiarazioni di tre variabili: **CONN** sarà la connessione al database da aprire, **RS** sarà il Recordset che verrà aperto dalla connessione ed i cui dati saranno restituiti alla chiamata della pagina ASP; **XMLStream** è invece uno stream che verrà utilizzato per trasformare il Recordset in XML da inviare al browser.

```
14.   On Error Resume Next
15.   Set RS = Server.CreateObject("ADODB.Recordset")
16.   Set CONN = Server.CreateObject("ADODB.Connection")
17.   CONN.ConnectionString = "Driver={Microsoft Access Driver (*.mdb)}; DBQ=" &
    Server.MapPath(NOMEDB)
18.   CONN.Open
19.   RS.Open "SELECT * FROM " & Request.QueryString("TableName"), CONN, adOpenkeyset,
    adLockPessimistic
```

È necessario dedicare una nota particolare alla possibilità che si verifichi un errore durante l'apertura del Recordset richiesto. Dall'altro lato dell'applicazione si trova un programmino Visual Basic, che si aspetta comunque dei dati in una forma corretta. Sarà pertanto necessario verificare la presenza di errori ed in questo caso restituire comunque un Recordset. Ecco perché alla riga 14 abbiamo inserito la funzione di gestione degli errori.

Istanziate le due variabili **CONN** ed **RS**, segue la normale sequenza di istruzioni per l'apertura di una connessione ADO e del Recordset richiesto (righe 17-19).

```
20.   If Err Then
21.     If RS Is Nothing Then Set RS = Server.CreateObject("ADODB.Recordset")
22.     With RS
23.       If .State = adStateOpen Then .Close
24.       .Fields.Append "Type", adVarChar, 9
25.       .Fields.Append "Source", adVarChar, 50
26.       .Fields.Append "Description", adVarChar, 255
27.       .Fields.Append "Code", adInteger
28.       .Source = ""
29.       .ActiveConnection = Nothing
30.       .Open
31.       .AddNew
32.       .Fields("Type") = "##Error##"
33.       .Fields("Source")= Err.Source
34.       .Fields("Description") = Server.URLEncode(Err.Description)
35.       .Fields("Code") = Err.Number
36.       .Update
37.     End With
38.   End If
39.   On Error Goto 0
```

Nel caso che si verifichi un errore, faremo in modo che la pagina restituisca comunque un Recordset valido contenente il messaggio di errore generato.

Alla riga 21 viene istanziato il Recordset se la precedente istanza ha avuto esito negativo. Se il Recordset **RS** arriva aperto, sarà chiuso per permettere la modifica della struttura.

Saranno aggiunti quattro campi (*Type*, *Source*, *Description* e *Code*), viene aperto il Recordset, vengono aggiunti dei dati che indicano l'errore che si è verificato (righe 24-36).

 Copyright Visual Basic Simple 2002

Solo allora viene rimpossa la funzione di gestione degli errori. Dalla riga 39 in poi tutti gli errori che si saranno verificati non saranno controllati. L'ideale sarebbe di creare una nuova sezione di codice per la gestione dei nuovi errori.

```
40.    Set XMLStream = Server.CreateObject("ADODB.Stream")
41.    RS.Save XMLStream, adPersistXML
42.    RS.Close
43.    CONN.Close
44.    Set RS = Nothing
45.    Set CONN = Nothing
46.    Response.ContentType = "text/xml"
47.    Response.Write XMLStream.ReadText
48.    XMLStream.Close
49.    Set XMLStream = Nothing
50. %>
```

Alla riga 40 viene istanziato lo Stream. Alla riga successiva viene salvato il Recordset RS all'interno dello stream XMLStream; la costante adPersistXML identifica il tipo di salvataggio (in XML quindi). Alle righe 42-45 vengono chiusi e deallocati Recordset e connessione.

Completiamo il nostro programmino ASP con la restituzione dei dati al chiamante. Viene innanzitutto definito il tipo di dati restituiti (riga 46) e poi viene restituito il contenuto di XMLStream, prima di chiuderlo e deallocarlo.

Se richiamassimo questa pagina ASP mediante un browser riceveremmo un Recordset XML da esplorare. Nel nostro caso abbiamo invece un semplice programma scritto in Visual Basic; vediamolo:

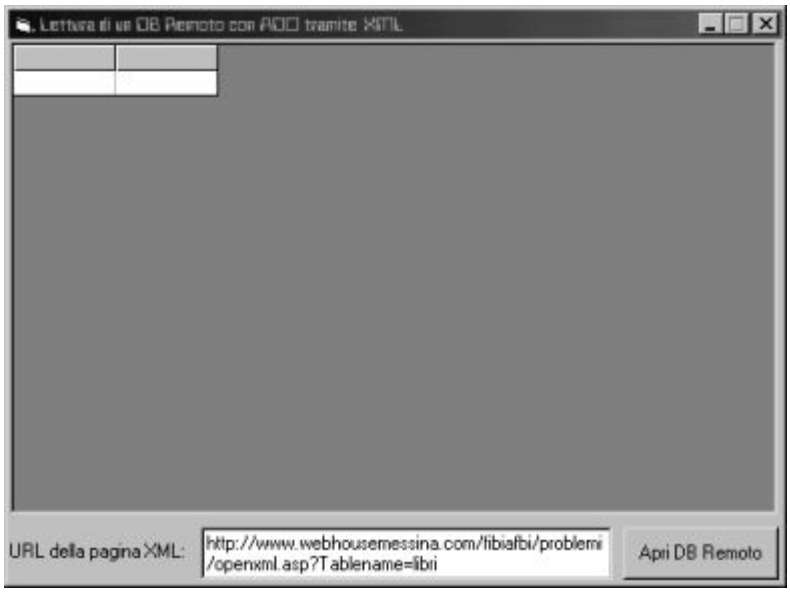


Figura 1

Il progetto farà uso di *ADO 2.6* e pertanto sarà necessario aggiungere tale libreria alla lista dei riferimenti del progetto. Il form contiene solo 4 controlli. Il primo di questi è una ***Microsoft FlexGrid*** di nome **Griglia**, che occupa quasi la totalità della superficie del form. Gli altri controlli sono una ***Label*** di nome **URLLabel**, una ***TextBox*** di nome **URLText** ed un ***CommandButton*** di nome **ApriDBRemoto**.
Il codice è molto semplice:

 Copyright Visual Basic Simple 2002

```
 1. Option Explicit
 2.
 3. Private Sub ApriDBRemoto_Click()
 4.   Dim RS As ADODB.Recordset
 5.   Dim Y As Integer
 6.   Dim X As Integer
 7.
 8.   On Error GoTo ERRORE
 9.   Set RS = New ADODB.Recordset
10.   RS.Open URLXML.Text
```

Nel momento in cui l'utente clicca sul pulsante ApriDBRemoto viene avviata la connessione al sito Internet remoto. Il suo funzionamento è molto semplice.

Alla riga 4 dichiariamo un *Recordset* di nome **RS** e lo allochiamo alla riga 9. Tutti gli errori saranno gestiti (riga 8) dalla sezione di codice identificata con l'etichetta **ERRORE**.

Alla riga 10 viene effettuata l'apertura del Recordset, fornendo come origine della tabella l'indirizzo Internet presente nella TextBox **URLXML**.
La pagina chiamata dovrà restituire un Recordset XML altrimenti verrà generato un errore.

```
11.   If (RS.Fields(0).Name = "Type") And (RS.Fields(1).Name = "Source") Then Err.Raise
    RS.Fields("CODE").Value, RS.Fields("SOURCE").Value, RS.Fields("DESCRIPTION").Value
12.   Griglia.Clear
13.   Griglia.Cols = RS.Fields.Count
14.   Griglia.Rows = RS.RecordCount + 1
15.   For X = 1 To RS.RecordCount
16.     For Y = 0 To RS.Fields.Count - 1
17.       If X = 1 Then Griglia.TextMatrix(0, Y) = RS.Fields(Y).Name
18.       Griglia.TextMatrix(X, Y) = RS.Fields(Y).Value & ""
19.     Next Y
20.     RS.MoveNext
21.   Next
22.   RS.Close
23.   Set RS = Nothing
24.   Exit Sub
```

Ricordiamoci cosa faceva la nostra pagina ASP nel momento in cui si verificava un errore! Essa creava un Recordset contenente quattro campi e li riempiva con i dati dell'errore generato.

Pertanto alla riga 11 viene verificato se i primi due campi del Recordset ricevuto si chiamano **Type** e **Source**. In tal caso viene generato volontariamente un errore con i dati presenti nel Recordset. L'errore sarà gestito dalla funzione apposita.

Solo allora sarà possibile azzerare la **Griglia** e riempirla con i dati ricevuti. Dopo aver preparato righe e colonne, verranno utilizzati due cicli per estrarre i dati dal Recordset (righe 12-21).
Per la prima riga soltanto, sarà eseguita un'istruzione aggiuntiva, che riempie oltre le celle della griglia, anche le intestazioni della colonna (riga 17).

La procedura si conclude con la chiusura e deallocazione del Recordset **RS**.

```
25. ERRORE:
26.   MsgBox "Si è verificato un errore." & vbNewLine & Err.Description, vbCritical +
    vbOKOnly, "DB Remoto"
27.   If RS.State = adStateOpen Then RS.Close
28.   Set RS = Nothing
29. End Sub
```

 Copyright Visual Basic Simple 2002

```
30.
```

Se si dovesse verificare un errore sarà mostrata una finestra di dialogo con la descrizione dell'errore (riga 26) e se il Recordset è ancora aperto sarà chiuso prima di essere deallocato.

```
31. Private Sub Form_Unload(Cancel As Integer)
32.   MsgBox "Il presente codice è stato sviluppato sulla base di un codice di Moreno
    Sirri" & vbNewLine & "Moreno Sirri VB Site - http://www.msvbsite.net",
    vbInformation + vbOKOnly, Me.Caption
33. End Sub
34.
35. Private Sub Griglia_DblClick()
36.   Griglia.Clear
37. End Sub
38.
39. Private Sub URLXML_KeyPress(KeyAscii As Integer)
40.   If KeyAscii = 13 Then KeyAscii = 0
41. End Sub
```

Seguono tre semplicissime funzioni non fondamentali, ma utili. La prima mostra un avviso sull'autore del codice originale, per rispettare le regole di copyright presenti nella pagina da cui viene fatto spunto per questo esempio (riga 31-33).

Nel momento in cui l'utente clicca due volte sulla griglia essa sarà azzerata. La sua funzione si rivela utile prima di richiedere una nuova pagina, per verificare se i dati sono stati effettivamente aggiornati (righe 35-37).

L'ultima funzione blocca la pressione del tasto Invio nella casella URLXML, in modo da non generare un indirizzo Internet con un carattere di Invio inserito in mezzo (righe 39-41).

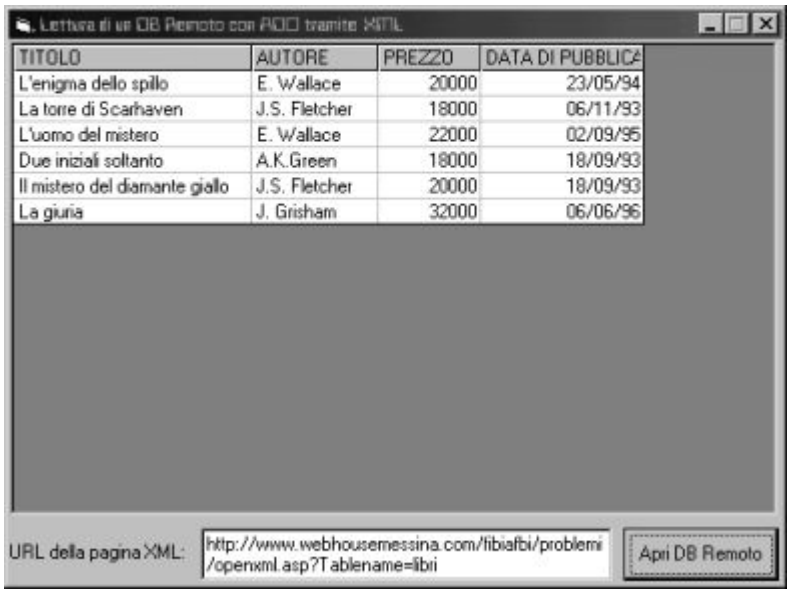


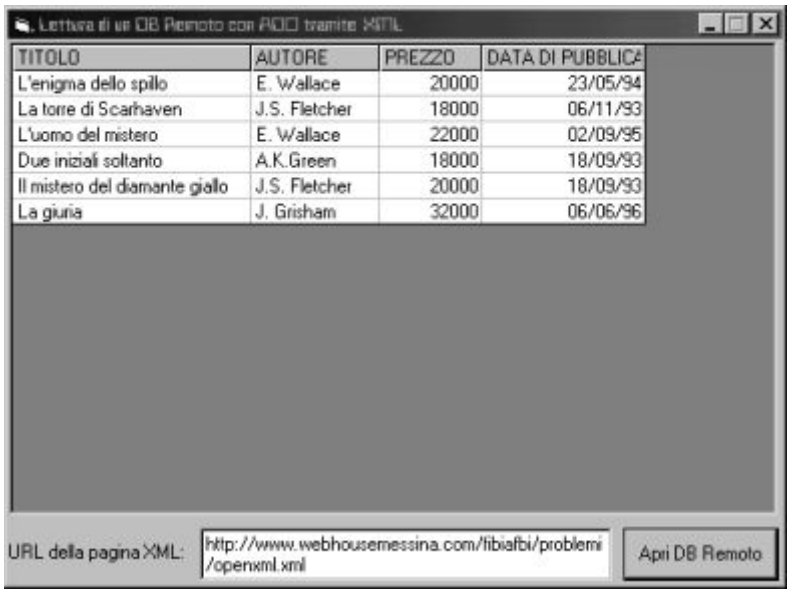
Figura 2

Possiamo passare alla prova del programma, molto semplice.
Abbiamo richiamato la nostra pagina ASP richiedendo l'indirizzo della pagina seguito dal parametro **`Tablename=libri`**. Essa richiederà l'apertura della tabella **LIBRI** che sarà mostrata nella griglia.

 Copyright Visual Basic Simple 2002

Se si dovesse generare un errore, sarà mostrato un avviso di errore che, purtroppo, conterrà tantissimi caratteri incomprensibili a causa della diversa codifica tra Visual Basic e Internet.

È molto interessante notare che è possibile anche salvare i Recordset XML in files con estensione XML e poi richiamarli fornendo semplicemente l'indirizzo Internet della pagina XML salvata.



Questa soluzione si rivela molto utile per alleggerire il lavoro del server e per poter utilizzare anche connessioni a database remoti quando il server ASP è inutilizzabile o non disponibile. Sarà possibile infatti inserire pagine XML statiche anche in server che non supportano ASP.

Il progetto è soltanto la punta di un grosso iceberg e mediante soluzioni di questo genere è possibile creare anche applicazioni di una certa grandezza ed importanza. In questo esempio è stata effettuata la sola lettura del Recordset originale ma con questa stessa tecnica è possible anche effettuare modifiche di dati sul server remoto.

La gestione degli errori è molto rozza e presenta numerose pecche.
Inoltre, la diversa codifica tra Internet e Visual Basic fa sì che non sia possible restituire Recordset contenenti caratteri accentati o altri simboli senza effettuare la codifica Internet. Tale codifica renderà però i dati illeggibili al client a meno di sviluppare una funzione di decodifica (non molto complessa comunque).

Moreno Sirri e Fibia FBI
Basato su: Gestione di un Database Remoto
18 Giugno 2001

Torna all'introduzione delle Richieste dei lettori

 Copyright Visual Basic Simple 2002