Esportare dati da AS/400 in un foglio Excel

http://www.vbsimple.net/howto/ht_051.htm

(Articolo temporaneo in attesa di essere trasferito ad altra sezione)

Difficoltà: 3 / 5

Un problema che affligge parecchi programmatori è quello di far comunicare un programma scritto in Visual Basic con un **AS/400**, richiedere l'esportazione di certi dati dal server e reimportarli, ad esempio, in un foglio *Excel* per semplificarne l'utilizzo sul lato client.

In questo articolo sarà spiegato come stabilire una connessione all'AS/400 utilizzando il driver ODBC fornito con *IBM Client Access* e *Client Access Express*.
A tal scopo è necessario creare un **DSN** (*Data Source Name*) utente o di sistema al database posto sul server. Il DSN può essere creato utilizzando il pannello di amministrazione ODBC oppure utilizzando le funzioni API ODBC. La connessione sarà effettuata mediante i modelli dati DAO e ADO.

L'accesso al foglio Excel richiede l'aggiunta di un riferimento alla *Microsoft Excel 8.0 Object Library* (EXCEL8.OLB) in caso di Excel 97 e *Microsoft Excel 9.0 Object Library* (EXCEL9.OLB) in caso di Excel 2000. La cartella Excel si chiamerà **CARTEL1.XLS** e sarà contenuta nella cartella del programma. Nel nostro esempio il server si chiama **AS400** ed i dati da estrarre sono contenuti nella tabella **NOMINATIVI** contenuta nel database posto sull'AS/400 nella libreria **CLIENTI**. L'utente che accederà al server si chiamerà **UTENTE** e la password associata sarà **PASSWORD**. Il DSN utilizzato sarà chiamato **DSNAS400** ed utilizzerà il driver *Client Access ODBC Driver (32-bit)*.

La prima delle due soluzioni utilizza il modello dati più vecchio: Data Access Object (DAO) ed a tal scopo è necessario aggiungere un riferimento alla libreria *Microsoft DAO x.xx Object Library* (DAOxx.DLL). Questa soluzione è necessaria se si utilizza Microsoft Excel 97. Vediamo il codice del progetto:

```
1. Sub EstraiDati(ByVal strSQL As String, ByVal strWbk As String)
2.     Dim wrkODBC As DAO.Workspace
3.     Dim conn As DAO.Connection
4.     Dim rs As DAO.Recordset
5.     Dim xlApp As Excel.Application
6.     Dim xlQT As Excel.QueryTable
7.     Dim xlWS As Excel.Worksheet
8.
```

La routine che estrae i dati e li inserisce nel foglio Excel è chiamata **EstraiDati** e riceve due argomenti: **strSQL** è una stringa SQL, utilizzata per estrarre tutta o parte di una o più tabelle; **strWbk** indica invece il percorso della cartella Excel.

La funzione utilizza parecchi riferimenti ad oggetti: **wrkODBC** è l'area di lavoro

ODBCDirect, **conn** è la connessione basata sull'area di lavoro, mentre **rs** è il Recordset che conterrà i dati.
Seguono gli oggetti Excel: **xlApp** indica l'applicazione Microsoft Excel, **xlQT** è la QueryTable che riceverà i dati del Recordset e che verrà aggiunta al foglio di lavoro **xlWS**.

```
 9.     Set wrkODBC = CreateWorkspace("ODBCAS400", "UTENTE", "PASSWORD", dbUseODBC)
10.     Set conn = wrkODBC.OpenConnection("Conn1", dbDriverComplete, ,
    "ODBC;DSN=DSNAS400;DATABASE=AS400.CLIENTI")
11.     Set rs = conn.OpenRecordset(strSQL)
12.
```

La prima operazione da eseguire sarà quella di stabilire la connessione alla fonte dati ODBC mediante DSN, ma prima di poter accedere ad una qualsiasi fonte dati ODBC è necessario creare un'area di lavoro *ODBCDirect*; in seguito sarà aperta la connessione ed estratti i dati da essa.

Alla riga 9 è creato una nuova area di lavoro (Workspace) di nome **ODBCAS400**, ma il nome poteva essere qualunque altro valore; il proprietario dell'area di lavoro è **UTENTE** ed utilizza la password **PASSWORD** per accedere alle fonti dati. L'ultimo parametro ***dbUseODBC*** specifica che il Workspace da creare è di tipo *ODBCDirect*, opposto al tipo standard Jet. La nuova area di lavoro è quindi contenuta nella variabile oggetto **wrkODBC**.

Alla riga successiva è avviata la connessione al server; il nome della connessione è Conn1 ma ciò non è rilevante al nostro scopo; il secondo argomento ***dbDriverComplete*** specifica che nel caso venissero forniti i parametri di connessione errati, sarà proposta la finestra di accesso del driver utilizzato. Il valore è contenuto nell'enumerazione **DriverPromptEnum** e può assumere uno dei seguenti valori:

- dbDriverComplete
  Determina la visualizzazione della finestra di accesso del relativo driver nel caso che le informazioni fornite nella stringa di connessioni siano insufficienti.

- dbDriverNoPrompt
  Determina l'uso esclusivo della stringa di connessione. Se questa è errata o incompleta sarà generato un errore.

- dbDriverPrompt
  Determina la possibilità di scelta fra un elenco di DSN disponibili nel sistema e quindi richiama la finestra di accesso del relativo driver.

- dbDriverCompleteRequired
  Simile al valore precedente ma nasconde tutte le informazioni non strettamente necessarie.

Il terzo argomento consente di specificare un valore che indica se la connessione è effettuata in modalità sola lettura oppure siano consentite le modifiche. Il valore è ignorato in questo articolo.

L'ultimo argomento della funzione OpenConnection specifica la stringa di connessione da utilizzare per accedere alla fonte dati. Il primo valore (**ODBC**) specifica il driver da

utilizzare e nel nostro caso fa riferimento all'insieme dei DSN; le informazioni che seguono indicano il nome del DSN (**DSNAS400**) ed il database utilizzato (**AS400.CLIENTI**). Non sono stati volutamente specificati nome utente e password perché tali valori saranno quindi ereditati dall'oggetto Workspace. In caso contrario sarebbe stato necessario specificare ***UID=UTENTE;PWD=PASSWORD***.

Infine alla riga 11 sarà recuperato il Recordset con i dati desiderati dalla connessione **conn**. I dati da recuperare dipendono dalla variabile **strSQL**.

```
13.     Set xlApp = New Excel.Application
14.     With xlApp
15.         .Visible = True
16.         .Workbooks.Open filename:=strWbk
17.         Set xlWS = .Sheets(1)
18.         Set xlQT = xlWS.QueryTables.Add(rs, .Range("A1"))
19.     End With
20.     xlQT.Name = "DB"
21.     xlQT.Refresh
22.
```

Alla riga 13 è richiamata l'applicazione Excel cosicché l'oggetto **xlApp** faccia riferimento alla nuova istanza di Microsoft Excel. In sequenza l'applicazione è mostrata, viene aperta la cartella di lavoro specificata nell'argomento di funzione **strWbk**, è scelto il primo foglio ed a questo viene aggiunta una nuova *QueryTable* contenente il Recordset **rs**, alla posizione A1 del foglio di lavoro. Il ***Refresh*** alla riga 21 fa sì che i dati della *QueryTable* siano riportati nel foglio di lavoro.

```
23.     xlApp.Workbooks(1).Save
24.     xlApp.Workbooks.Close
25.     xlApp.Quit
26.     rs.Close
27.     conn.Close
28.     wrkODBC.Close
29.     Set rs = Nothing
30.     Set conn = Nothing
31.     Set wrkODBC = Nothing
32.     Set xlQT = Nothing
33.     Set xlWS = Nothing
34.     Set xlApp = Nothing
35. End Sub
```

Segue infine la fase di chiusura e pulizia della memoria. Tutti gli oggetti aperti sono chiusi e deallocati. La cartella di lavoro Excel prima della chiusura viene salvata (riga 23).

La seconda soluzione utilizza invece il nuovo modello dati: ActiveX Data Objects (ADO) e stabilire una connessione risulta ben più semplice. A tal scopo è necessario aggiungere un riferimento a Microsoft ActiveX Data Objects 2.x Library (MSADO15.DLL). Questa soluzione è necessaria se si utilizza Microsoft Excel 2000/XP. La stessa routine EstraiDati può essere riscritta come segue:

```
1. Sub EstraiDati(ByVal strSQL As String, ByVal strWbk As String)
2.     Dim conn As ADODB.Connection
3.     Dim rs As ADODB.Recordset
4.     Dim xlApp As Excel.Application
5.     Dim xlQT As Excel.QueryTable
6.     Dim xlWS As Excel.Worksheet
7.
```


Rispetto la versione precedente è stato eliminato l'oggetto Workspace, mentre gli oggetti **conn** e **rs** cambiano la loro libreria da DAO ad ADODB. Sono mantenuti intatti i riferimenti agli oggetti Excel.

```
 8.      Set conn = New ADODB.Connection
 9.      conn.Open "DSN=DSNAS400;DATABASE=AS400.CLIENTI", "UTENTE", "PASSWORD"
10.      Set rs = New ADODB.Recordset
11.      rs.CursorLocation = adUseClient
12.      rs.Open strSQL, conn, adOpenStatic, adLockReadOnly
13.
```

Alla riga 8 e 10 sono allocati rispettivamente una nuova connessione ed un nuovo Recordset. La connessione al DSN è aperta alla riga 9: tutto ciò che è necessario fornire è il nome del DSN, l'utente e la sua password associata; il nome del database a cui collegarsi è necessario solo se si utilizza un database differente da quello predefinito per l'utente.

Prima di poter estrarre i dati tramite Recordset è fondamentale cambiare la posizione del cursore dati da server a client. In caso contrario si genererà un errore (vedi l'articolo Q263498 della Microsoft Knowledge Base). La riga 12 richiama il metodo **Open** del Recordset con numerose informazioni: il primo argomento specifica la query SQL utilizzata per estrarre i dati; il secondo argomento è indica la connessione cui l'oggetto **rs** fa riferimento; il terzo argomento specifica il tipo di cursore dati da utilizzare; l'ultimo argomento specifica invece il tipo di blocco da attivare. Si rimanda alla Sezione Database per un'approfondimento di questi concetti.

```
14.      Set xlApp = New Excel.Application
15.      With xlApp
16.          .Visible = True
17.          .Workbooks.Open filename:=strWbk
18.          Set xlWS = .Sheets(1)
19.          Set xlQT = xlWS.QueryTables.Add(rs, .Range("A1"))
20.      End With
21.      xlQT.Name = "DB"
22.      xlQT.Refresh
23.
```

Il segmento di codice per l'automazione di Microsoft Excel è identico a quello utilizzato nella soluzione precedente. Si vuole soltanto sottolineare che il metodo Add dell'insieme QueryTables richiede il passaggio di un oggetto Recordset, così come la soluzione precedente. La differenza sta nella classe di questo oggetto: sono rifiutati i Recordset DAO ma sono ammessi soltanto quelli di tipo ADODB.

```
24.      xlApp.Workbooks(1).Save
25.      xlApp.Workbooks.Close
26.      xlApp.Quit
27.      rs.Close
28.      conn.Close
29.      Set rs = Nothing
30.      Set conn = Nothing
31.      Set xlQT = Nothing
32.      Set xlWS = Nothing
33.      Set xlApp = Nothing
34.  End Sub
```

Segue la chiusura e la deallocazione di tutti gli oggetti utilizzati. La cartella di lavoro è salvata prima di chiudere l'istanza **xlApp** di Microsoft Excel.

La routine per l'esportazione dei dati può essere richiamata con una semplice riga:

```
EstraiDati "SELECT * FROM NOMINATIVI", App.Path & "\CARTEL1.XLS"
```

La possibilità di automatizzare Microsoft Excel e recuperare fonti dati remote per utilizzarle con uno strumento esterno può semplificare numerosi problemi.

Fibia FBI e Marco Rubaltelli
16 Settembre 2002

 Torna all'indice degli HowTo

