Prof.ssa Sara Michelangeli

ADO.NET

L'oggetto DataReader :accesso e aggiornamento di database in modalità connessa

ADO.NET è il componente per l'accesso ai dati del Microsoft .NET Framework, è una evoluzione del precedente componente ADO di VB6 , con il quale tuttavia ci sono pochi punti in comune e nessuna compatiblità

ADO.NET, mette a disposizione una serie di classi che consentono l'accesso a differenti tipi di database. Essendo parte integrante del .NET Framework e costituite da codice managed, forniscono la massima efficienza possibile

Le classi si trovano in una serie di namespace, tra i quali:

- **System.Data** : contiene gli oggetti di base della classe ADO.NET utilizzati per accedere e memorizzare i dati;
- System.Data.Common : contiene le classi di base utilizzate da altri oggetti;
- **System.Data.Odbc : contiene le classi pubbliche per connettersi/manipolare attraverso driver ODBC;**
- System.Data.OleDb : contiene le classi pubbliche utilizzate per connettersi/manipolare attraverso un provider OLE.DB, in passato molto usato da Microsoft, oggi in via di dismissione.

Tra i Data Consumers, cioè i destinatari dei servizi ADO ci sono tipicamente **winform**, come nella nostra applicazione e webform.

Il Data Provider, Managed Provider consente la **connessione con la sorgente dati fisica**, è quindi specifico per data sourcers :noi ci riferiamo al l'**ODBC** Data Provider in quanto, il data source OLEDB, fino ad oggi tipico dell'ambiente Microsoft, verrà dismesso entro il 2018. E' da notare comunque che i providers contengono gli stessi oggetti seppur con alcune differenze per nome, proprietà e metodi.

Un Data Provider contiene l'oggetto **Connection**, l'oggetto **Command**, l'oggetto **DataAdapter** e l'oggetto **DataReader.**

Nell'oggetto **Connection, connessione fisica verso la sorgente dati**, vi sono i metodi che consentono aprire e chiudere una connessione, cambiare il database, utilizzare le transazioni.

Per gestire i dati vengono utilizzate due modalità diverse, a seconda se si lavori a diretta comunicazione con la sorgente dei dati o in modalità disconnessa.

In questa lezione esaminiamo la modalità connessa, mentre l'altra modalità di lavoro, quella disconnessa, basata sull'oggetto **DataSet (**copia dell'istanza della sorgente dati,) che non ha relazione con la fonte dei dati, ma li gestisce grazie al DataAdapter,
viene esaminata in un'altra lezione.

L'oggetto **DataReader** è stato progettato per comunicare direttamente con l'origine dati.

Essendo di dimensioni ridotte è molto efficiente e, considerando l'accesso diretto al cursore consente di esaminare i risultati in un modo estremamente rapido, , in modalità di sola lettura.

In questa lezione su **ADO.NET** impariamo a gestire il database con riferimento al Provider ODBC e in modalità connessa, quindi con riferimento diretto alla sorgente fisica dei dati utilizzando

l'oggetto **DataReader** e le istruzioni del linguaggio SQL, per l'accesso ai dati (QL) e per l'effettivo aggiornamento (DML).

Sappiamo di aver bisogno di un oggetto **Connection** che ci consenta di realizzare la connessione fisica verso la sorgente dati.

Dopo aver inserito da progetto riferimento .NET **System.Data**
o da codice Imports System.Data

dichiariamo nel modulo

```
Public con As New Odbc.OdbcConnection 'per la connessione
```

e nel load del form di avvio impostiamo la stringa di connessione con il database che vogliamo utilizzare e apriamo la connessione

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

        con.ConnectionString = "Driver={Microsoft Access Driver (*.mdb)};DBQ=nomedb.mdb;"
        con.Open()

    End Sub
```

Una volta aperta la connessione, per svolgere il compito di acquisire e gestire i dati dobbiamo fare riferimento ad altre componenti.

Accesso ai dati

L'oggetto Connection infatti possiede solo alcuni metodi di base fondamentali per **aprire** e **chiudere** la connessione, mentre per recuperare effettivamente i dati dobbiamo gestire:

- un'**istruzione SQL** che selezioni le informazioni da prelevare
- un **oggetto Command** che esegua l'istruzione SQL
- un oggetto **DataReader** per rendere disponibili le informazioni recuperate

L'**oggetto Command** rappresenta una istruzione SQL da eseguire sulla sorgente dati. Per utilizzare un command occorre

- dichiararlo e istanziarlo
- definire il contenuto della istruzione SQL da utilizzare
- associare al command una connessione aperta e disponibile
- associare al command l'istruzione SQL

Naturalmente, prima di iniziare occorre verificare di aver eseguito correttamente tutto il codice di apertura della connessione.

Facendo riferimento al nostro database con la tabella agenda ed i campi nick ed email, esaminiamo i passi di stesura del codice . Occorre

- **dichiarare ed istanziare il Command**

```
Public mycmd As New Odbc.OdbcCommand
```

- **definire il contenuto della istruzione SQL da utilizzare**

```
Dim sql As String
sql = "select  agenda.nick, agenda.email from agenda;"
```

- **associare al command una connessione aperta e disponibile**

```
mycmd.Connection = con
```

- **associare al command l'istruzione SQL**

```
 mycmd.CommandText = sql
```

e' possibile utilizzare anche la sintassi

```
Dim mycmd As New Odbc.OdbcCommand(sql, con)
```

direttamente in fase di dichiarazione e istanziazione dell'oggetto command

Una volta definito il command occorre scegliere la modalità di accesso, connessa o disconnessa. Come modalità di lavoro in questa lezione utilizziamo il Command con un **DataReader**, utilizzando una **connessione diretta**.

L'oggetto DataReader infatti è stato progettato per comunicare direttamente con l'origine dati. Consente di recuperare velocemente le informazioni e di esaminare i risultati, **in modalità di sola lettura**, in un modo estremamente rapido , supporta un accesso fast forward-only. Gestendo una connessione diretta, va richiuso ad ogni utilizzo.
Per creare un DataReader si usa il metodo dell'oggetto Command, dopo aver impostato il codice illustrato in precedenza.

```
Dim myDataReader As Odbc.OdbcDataReader
myDataReader = mycmd.ExecuteReader()
```

A questo punto le informazioni selezionate dall'istruzione sql sono disponibili e possono essere recuperate, per singola riga, con l'istruzione **myDataReader.read()** e per i singoli campi con la sintassi myDataReader("nomecampo") e gestiti con ordinari controlli ed istruzioni di visual basic.
Ecco ad esempio una stampa dell'intera tabella in listbox

```
Public Class Form
    Public mycmd As New Odbc.OdbcCommand
    Private Sub Btnsql_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Btnsql.Click
        Dim myDatareader As Odbc.OdbcDataReader
        Dim sql As String
        sql = "select  agenda.nick, email from agenda;"
        mycmd.Connection = con
        mycmd.CommandText = sql
        'o in alternativa
```

```
        'Dim mycmd As New Odbc.OdbcCommand(sql, con)
        myDatareader = mycmd.ExecuteReader()
        'ciclo di stampa in listbox
        While myDatareader.Read()
            Lstagenda.Items.Add(myDatareader("nick") + "   " + myDatareader("email"))
        End While
        myDatareader.Close()
    End Sub
```

L'istruzione SQL che costruisce la query di selezione, puo' essere anche risultato di join tra due o più tabelle. Supponendo di avere nel database anche una tabella chiamata login con nick e pw è possibile impostare una query di selezione che tenga conto degli opportuni vincoli interrelazionali

```
  Private Sub Btnsql_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Btnsql.Click

        Dim myDatareader As Odbc.OdbcDataReader
        Dim sql As String
        sql = "select  agenda.nick, email,pw from agenda,login where
agenda.nick=login.nick;"
        'Dim mycmd As New Odbc.OdbcCommand(sql, con)
        mycmd.Connection = con
        mycmd.CommandText = sql
        myDatareader = mycmd.ExecuteReader()
        While myDatareader.Read()
        Lstagenda.Items.Add(myDatareader("nick")+ "   "+ myDatareader("email")+ "   " + myDatareader("pw"))
        End While
        myDatareader.Close()
    End Sub
```

E' inoltre possibile costruire query parametriche anche con dati acquisiti da input. Nell'esempio seguente viene stampata l'email di un nick scelto da input

```
 Private Sub btncercanick_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btncercanick.Click
        Dim mydatareader As Odbc.OdbcDataReader
        Dim sql As String
        sql = "select nick, email from agenda where nick='" + Txtnick.Text + "';"
        mycmd.Connection = con
        mycmd.CommandText = sql
        mydatareader = mycmd.ExecuteReader()
        mydatareader.Read()
        txtemail.text=mydatareader("email"))
        mydatareader.Close()

    End Sub
```

Manipolazione delle Tabelle (DML)

Per eseguire un'istruzione SQL di **manipolazione delle tabelle (DML),** dopo aver aperto la connessione, e costruita l'opportuna stringa in sql (insert, delete, update), si inizializza il command

```
Dim cmd As New Odbc.OdbcCommand(sql, con)
```

con la connessione e l'istruzione sql

e si utilizza la sintassi

```
cmd.ExecuteNonQuery()
```

come negli esempi seguenti

```
Private Sub Btnins_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Btnins.Click

        Dim sql As String
        sql = "insert into agenda (nick, email) values ('" + Txtnick.Text + "','" +
Txtemail.Text + "')"
        Dim cmd As New Odbc.OdbcCommand(sql, con)
        cmd.ExecuteNonQuery()
    End Sub


    Private Sub Btnmodifica_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Btnmodifica.Click
        Dim sql As String
        sql = "update agenda set email='" + Txtemail.Text + "'where nick= '"+Txtnick.Text
+ "'"
        Dim cmd As New Odbc.OdbcCommand(sql, con)
        cmd.ExecuteNonQuery()

    End Sub

    Private Sub Btncancella_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Btncancella.Click
        Dim sql As String
        sql = "delete from agenda where nick= '" + Txtnick.Text + "'"
        Dim cmd As New Odbc.OdbcCommand(sql, con)
        cmd.ExecuteNonQuery()
    End Sub
```