Tecnologie di Sviluppo per il Web

Programmazione su Basi di Dati: ADO.NET

versione 1.1

G. Mecca – mecca@unibas.it – Università della Basilicata

Sommario

- Introduzione
- Connessioni
- Comandi
- DataReader
- Gestione delle Connessioni
- Altri Tipi di Statement
- DataSet – Cenni

Introduzione

- La tecnologia ADO.NET
  - ⇨evoluzioni delle precedenti tecnologie per la gestione dei dati sviluppate dalla Microsoft: ADO, DAO ...
  - ⇨unifica i vari protocolli di comunicazione con il DBMS sviluppati dalla Microsoft: ODBC, OLEDB...
  - ⇨fortemente ottimizzata per l'uso di SQLServer

Introduzione

- Attenzione
  - ⇨la tecnologia cambia ma gli aspetti metodologici restano esattamente gli stessi
  - ⇨ADO.NET è una tecnologia molto più complessa di JDBC
  - ⇨ne vediamo per cominciare un sottoinsieme che è sostanzialmente equivalente al sottoinsieme visto di JDBC

Introduzione

- Il namespace di riferimento
  - ⇨System.Data
  - ⇨definisce tutte le principali interfacce
  - ⇨IDbConnection
  - ⇨IDbcommand
  - ⇨IDataReader (analoga a ResultSet)
- Gli altri namespace
  - ⇨implementano i diversi “provider di dati”

Introduzione

- Provider di dati ADO.NET
  - ⇨analogo del driver per JDBC
  - ⇨fornisce una implementazione per le interfacce di System.Data per uno specifico sistema
  - ⇨la piattaforma .NET fornisce vari provider sotto forma di namespace standard
  - ⇨altri provider possono essere scaricati dalla rete sotto forma di dll

Introduzione

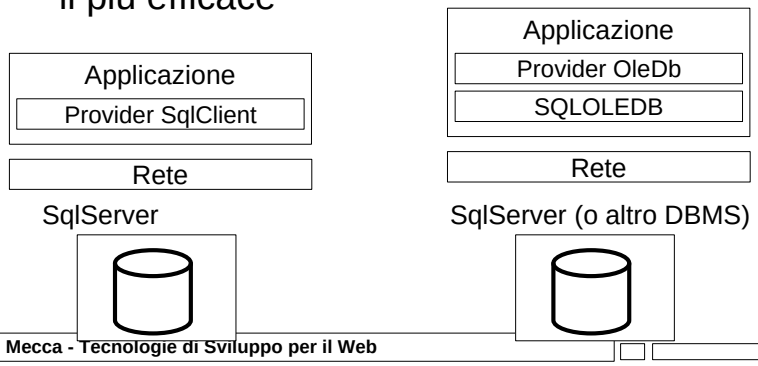
- I provider standard forniti
  - ⇨System.Data.SqlClient: provider nativo per SqlServer
  - ⇨System.Data.OracleClient: provider nativo per Oracle
  - ⇨System.Data.OleDb: provider di tipo “bridge” verso un provider OleDb (es: SQLOLEDB)
  - ⇨System.Data.Odbc: provider di tipo “bridge” verso un driver ODBC

Introduzione

- Un esempio di provider non standard
  - ⇨Npgsql, il provider di dati per PostgreSQL
  - ⇨distribuito sotto forma di file .zip sul sito gborg.postgresql.org
- Il package System.Data.Common
  - ⇨classi comuni a tutti i provider

Introduzione

- Il provider fondamentale
  - ⇨il provider nativo per SqlServer: di gran lunga il più efficace



Introduzione

- Le classi di System.Data.SqlClient
  - ⇨SqlConnection : IDbConnection
  - ⇨SqlCommand : IDbCommand
  - ⇨SqlDataReader : IDataReader
  - ⇨SqlException : System.Exception

Connessioni

- Per ottenere una connessione al DBMS
  - è necessario creare un oggetto di tipo IDbConnection utilizzando l'opportuna implementazione
  - specificando una stringa di connessione
- Stringa di connessione .NET
  - sequenza di coppie nome = valore separate da ; che descrivono i parametri di connessione

Connessioni

- Parametri per SqlConnection
  - Data Source: indirizzo IP del server (in alternativa: Server)
  - Database: nome della base di dati (in alternativa: Initial Catalog)
  - Integrated Security: true/false, specifica se utilizzare il meccanismo di aut. di Windows
  - User ID
  - Password

Connessioni

>> DataSource

- Per creare connessioni
  - ⇨un componente DataSource
  - ⇨il metodo GetConnection() inizializza la stringa di connessione sulla base di parametri fissati
  - ⇨crea l'oggetto di tipo SqlConnection
  - ⇨apre la connessione chiamando il metodo Open()
  - ⇨restituisce il rif. alla connessione aperta

```
namespace Unibas.Aci.Persistenza {
using System.Data;
public class DataSource {

   private string dataSource = "localhost";
   private string dataBase = "acibase";
   private string userId = "msdeuser";
   private string password = "msdeuser";

   public IDbConnection GetConnection() {
      IDbConnection connection = null;
      string connectionString = "Data Source = " + this.dataSource + "; " +
                                "Database = " + this.dataBase + "; " +   "Integrated Security = false; " +
                                "User ID = " + this.userId + "; " + "Password = " + this.password;
      try {
         connection = new System.Data.SqlClient.SqlConnection(connectionString);
         connection.Open();
      } catch (System.Data.SqlClient.SqlException sqle) {
         Close(connection);
         throw new DAOException("GetConnection: " + sqle);
      }
      return connection;
   }
   ...
}
```

Connessioni

- Nota
  - ⇨per creare connessioni utilizzando un provider diverso è sufficiente cambiare l'istruzione di creazione
  - ⇨è però in generale necessario anche cambiare la stringa di connessione
- Esempio: provider OleDb
  - ⇨è necessario aggiungere "Provider = SQLOLEDB" per l'accesso a SqlServer

Comandi

- Per effettuare operazioni sul DBMS
  - ⇨oggetti di tipo IDbCommand
  - ⇨implementati da SqlCommand
  - ⇨si ottengono dalla connessione utilizzando il metodo CreateCommand()
  - ⇨richiedono di impostare il valore di alcune proprietà, in particolare la proprietà CommandText

Comandi

>> DAOProprietario

- Due metodi principali
- ExecuteNonQuery()
  - ⇨esegue un comando che corrisponde ad un aggiornamento
- ExecuteReader()
  - ⇨esegue un comando che corrisponde ad una query SQL e restituisce un riferimento al DataReader che consente di scandire il risultato

DataReader

- Per scandire il risultato di una query
  - ⇨risultati di tipo IDataReader
  - ⇨implementati da SqlDataReader
  - ⇨vengono ottenuti dal metodo ExecuteReader() di IDbCommand
  - ⇨metodo per la scansione: boolean Read()
  - ⇨per l'accesso agli attributi si utilizza la sintassi delle mappe

DataReader

>> DAOProprietario

- Esempio: DoSelectProprietarioPerNome

```
connection = dataSource.GetConnection();
command = connection.CreateCommand();
string query = "select * from proprietari where nome = '" + nome + "'";
command.CommandText = query;
dataReader = command.ExecuteReader();
while (dataReader.Read()) {
   Proprietario proprietario = new Proprietario();
   proprietario.CodiceFiscale = (string)dataReader["codicefiscale"];
   proprietario.Nome = (string)dataReader["nome"];
   proprietario.CittaDiResidenza = (string)dataReader["cittadiresidenza"];
   proprietario.AnnoPatente = (int)dataReader["annoPatente"];
   listaProprietari.Add(proprietario);
}
```


Gestione delle Connessioni

- Chiusura di una connessione
  - ⇨metodo Close() di IDbConnection
- Il provider di SqlServer
  - ⇨gestisce un “pool di connessioni” standard
  - ⇨le connessioni non vengono chiuse ma restituite al pool
  - ⇨e successivamente riutilizzate

Altri Tipi di Statement

- Per utilizzare statement preparati
  - ⇨bisogna utilizzare una variante di IDbCommand
- Strategia
  - ⇨si specifica un command text con parametri; ciascun parametro è preceduto da @
  - ⇨viene preparato lo statement
  - ⇨successivamente per eseguirlo è sufficiente fornire un valore al parametro

Altri Tipi di Statement

- Una differenza rispetto a JDBC
  - ⇨è necessario creare esplicitamente un oggetto per ciascun parametro della query
  - ⇨specificando nome e tipo di dato
- I componenti necessari
  - ⇨interfaccia IDataParameter
  - ⇨implementazione SqlParameter

Altri Tipi di Statement

>> DAOProprietario

- Esempio: DAOUtente

```
connection = dataSource.GetConnection();
command = connection.CreateCommand();
string query = "select * from utenti where nomeutente = @nomeutente";
command.CommandText = query;
SqlParameter parametro = new SqlParameter();
parametro.ParameterName = "@nomeutente";
parametro.SqlDbType = SqlDbType.VarChar;
parametro.Size = 10;
command.Parameters.Add(parametro);
command.Prepare();
((SqlParameter)command.Parameters["@nomeutente"]).Value =
nomeUtente;
dataReader = command.ExecuteReader();
```


Altri Tipi di Statement

- Per le stored procedure
  - ⇨si utilizza comunque l'interfaccia IDBCommand
  - ⇨ma è possibile cambiare il tipo di comando attraverso la proprietà CommandType
- Esempio:
  - ⇨ IDbCommand command = null;
  - ⇨ command = connection.CreateCommand();
  - ⇨ command.CommandType = CommandType.StoredProcedure;

Altri Tipi di Statement

- Successivamente
  - ⇨è necessario specificare i parametri relativi alla chiamata della stored procedure
  - ⇨aggiungendoli alla collezione Parameters

DataSet – Cenni

- Attenzione
  - ⇨l'utilizzo esplicito dei comandi e dei DataReader è solo una delle modalità di utilizzo di ADO.NET
- In effetti
  - ⇨ADO.NET ha al suo interno un framework vero e proprio per la gestione della persistenza
  - ⇨basato sul concetto di DataSet

DataSet – Cenni

- DataSet
  - ⇨vista orientata agli oggetti su una porzione di una sorgente dati (tipicamente una base di dati)
- Caratteristiche del DataSet
  - ⇨viene utilizzata in modo disconnesso, ovvero non richiede una connessione permanente alla base di dati

DataSet – Cenni

- Utilizzo disconnesso
  - ⇨l'utente descrive dichiarativamente in un file di configurazione la struttura del DataSet
  - ⇨il sistema inizializza il DataSet
  - ⇨l'utente lavora sugli oggetti del DataSet
  - ⇨il sistema si preoccupa di tenere il DataSet aggiornato rispetto alla base di dati con una politica di aggiornamento periodico
  - ⇨questo garantisce ottime prestazioni

DataSet – Cenni

- Quindi
  - ⇨il programmatore inizializza il DataSet e poi lavora con gli oggetti relativi, senza doversi poi preoccupare delle modalità di sincronizzazione con la base di dati
  - ⇨in effetti sembra conseguire l'obiettivo di separare gli strati fornendo un meccanismo automatico per la gestione della persistenza

DataSet – Cenni

- Ma...
  - ⇨gli oggetti del DataSet non sono oggetti del modello (es: Proprietari e Automobili), ma oggetti che rappresentano le tabelle e le righe relative
  - ⇨quindi incoraggiano uno stile di programmazione in cui lo strato del modello non esiste

Riassumendo

- Introduzione
- Connessioni
- Comandi
- DataReader
- Gestione delle Connessioni
- Altri Tipi di Statement
- DataSet – Cenni

Termini della Licenza