Database locale in VS 2012+

Gestione di un file di database (localDB e Access) in Visual Studio 2012/13

Progetto di Informatica classe 5^{a}

Ambiente: .NET 4.5/C# 5.0

Anno 2013/2014

Indice generale

1 Introduzione

Il tutorial mostra come creare un file di database SQL Server e gestirlo all'interno di un progetto di Visual Studio. Successivamente, mostra come ottenere lo stesso risultato con un database Access.

1.1 Creazione database locale (localDB)

Un database SQL Server può essere gestito come un file separato, all'interno di un'istanza di LocalDB. Di fatto, in questo modo è possibile gestire un database SQL Server in modo simile a quello utilizzato per un database Access.

LocalDB

LocalDB è una versione "light" di SQL Server installata insieme a Visual Studio e utilizzata di default nella gestione di database SQL Server.

La procedura descritta di seguito mostra come creare un file di database LocalDB e gestirlo all'interno di un progetto.

1. Aggiungere al progetto un nuovo elemento di tipo "***service-based database***".

   Viene richiesto il nome del file (questo sarà il nome dato al file e non al database).

   Il file viene creato nella cartella del progetto.

   *Viene aggiunta una connessione (chiusa) su Server Explorer. Il nome della connessione è uguale al nome del file + estensione. Es: "**Anagrafica.mdf**".*

   *E' possibile modificare il nome della connessione. Es: "**Anagrafica**".*

2. Modificare la stringa di connessione su Server Explorer (mentre è chiusa).

   Eseguire il comando "**Modify connection...**" sulla connessione

   Nella dialog successiva cliccare sul bottone su "**Advanced...**"

   Impostare il nome del database alla voce **Initial Catalog** (preferibile utilizzare lo stesso nome del file senza estensione).

   La connessione viene aperta.

La procedura genera un file con estensione ".mdf" nella cartella di progetto e una connessione in Server Explorer che referenzia il file mediante il suo percorso assoluto. La compilazione del programma implica la copia del file nella cartella "Debug" (o Bin, in base al tipo di compilazione) contenente l'eseguibile dell'applicazione, denominata **cartella di output**.

Dopo la precedente procedura, viene creato un database su LocalDb (visibile con SQL Server Management Studio), che referenzia il file del progetto.

1.2 Inserimento nel progetto di un file di database esistente

Mediante il comando “Add | Existing item...” è possibile aggiungere un file di database precedentemente creato. (Nella dialog ricordarsi di impostare il filtro delle estensioni a “*.*”, altrimenti il file non sarà visibile.)

1.2.1 Copia&Incolla di un file di database esistente

E' sufficiente eseguire il comando copia sul file e successivamente eseguire il comando incolla in Visual Studio, a livello di progetto o di sottocartella.

1.3 Accesso alla stringa di connessione: Server Explorer

Selezionare la connessione su Server Explorer e accedere alle proprietà. Accedere alla proprietà “**Connection string**”.

La stringa di connessione è nella forma:

```
“Data Source=(LocalDB)\v11.0; AttachDbFilename="C:\...\nomefile.mdf";
Initial Catalog=nome-database; Integrated Security=True”
```

Nota bene: il percorso del file è assoluto ed è chiuso tra virgolette. Per utilizzare la stringa direttamente nel codice è necessario eliminare le virgolette.

Uso della stringa di connessione

La procedura appena descritta è opzionale. La stringa di connessione è, per l'appunto, soltanto una stringa, e dunque non ha importanza come venga generata. La cosa importante è che le parti “variabili”, nome database e nome file di database, siano corrette e facciano riferimento agli oggetti giusti.

Ottenere la stringa da Server Explorer garantisce semplicemente che questa sia scritta correttamente.

1.4 Memorizzare la stringa di connessione sul file di configurazione

Il file di configurazione, **App.Config**, consente di memorizzare delle informazioni allo scopo di renderle accessibili a tutta l'applicazione. In questo modo è possibile modificare alcune impostazioni del programma senza doverlo ricompilare.

Il file App.Config prevede una sezione, **ConnectionStrings**, nella quale collocare le stringhe di connessione. (Se la sezione non esiste, può essere creata). Ad esempio:

```
<connectionStrings>
    <add name="Northwind"
         connectionString="Data Source=(LocalDB)\v11.0;
                           AttachDbFilename=f:\...\Northwind.mdf;
                           Integrated Security=true"/>
  </connectionStrings>
```

La sezione ConnectionString prevede un o più tag **<add>**, ognuno delle quali ha due attributi di base: **name** e **connectionString**. Opzionalmente, può essere specificato l'attributo **providerName**:

```
<connectionStrings>
    <add name="DemoConsole.Biblioteca"
         connectionString="Data Source=(LocalDB)\v11.0;
                           AttachDbFilename=f:\...\Biblioteca.mdf;
                           Integrated Security=true"
         providerName="System.Data.SqlClient" />
</connectionStrings>
```

Provider name

Se presente, questo attributo definisce l'ADO.NET provider utilizzato per connettersi al database.

E' utile in quegli scenari nei quali la stringa di connessione viene utilizzata da Visual Studio o da eventuali framework di accesso ai dati, come Entity Framwork.

1.4.1 Accesso alla stringa di connessione memorizzata in App.Config

Per accedere da codice al file di configurazione occorre innanzitutto aggiungere un riferimento all'assembly **System.Configuration**, il quale definisce la classe statica **ConfigurationManager**. Occorre anche definire il namespace:

```
using System.Configuration;
```

Infine, per accedere alla stringa di connessione si usa ConfigurationManager:

```
var cnDef = ConfigurationManager.ConnectionStrings["Northwind"];
string cnStr = cnDef.ConnectionString;
```

Nota bene: il nome della connessione (attributo *name*) viene usato come chiave per accedere alla stringa.

1.5 Rendere il percorso del file relativo: uso di “DataDirectory”

Rendendo relativo il percorso del file si evita di dover modificare la stringa di connessione ogni qual volta si sposta il progetto. Per ottenere questo risultato si può usare la notazione “**|DataDirectory|**”.

```
“Data Source=(LocalDB)\v11.0; AttachDbFilename=|DataDirectory|\nomefile.mdf";
Initial Catalog=nome-database; Integrated Security=True”
```

Nelle applicazioni Web, DataDirectory viene sostituito con la cartella “App_Data”. Nelle Applicazioni Console e Windows, DataDirectory punta alla cartella contenente l'eseguibile, chiamata anche “**cartella di output”**.

La compilazione del programma provoca automaticamente la copia del file di database nella cartella di output. Ciò fa sì che il percorso “|DataDirectory|\<nomefile>” punti effettivamente al file di database, nonostante la versione originale del file si trovi in un altra cartella.

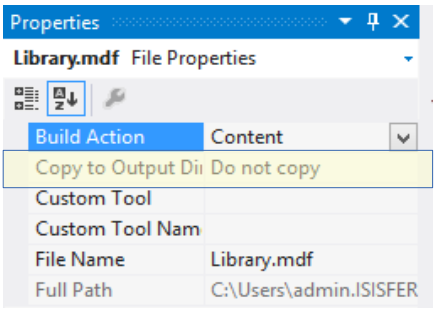
1.5.1 Evitare la copia del database nella cartella di output

La copia del file nella cartella di output può risultare utile se si tratta di un database Access (o SQLite), ma non con un database SQL Server.

Quest'ultimo impedisce la sovrascrittura di un file che è “attaccato” ad un'istanza (LocalDB, SQLEXPRESS, etc).

Come risultato di questo comportamento, l'accesso al database sarà possibile soltanto alla prima esecuzione del programma. Alla seconda esecuzione, la copia del file di database nella cartella di output provoca il tentativo di sovrascrivere un file esistente e già attaccato a LocalDB e produce un errore.

Per evitare la copia nella cartella di output occorre selezionare il file di database, accedere alla finestra “Proprietà” e impostare la voce “**Copy to Output...**” su “**Do not copy**”.



1.5.2 Accesso al database nella cartella del progetto.

La precedente impostazione invalida l'uso del percorso relativo utilizzato nella stringa di connessione (“DataDirectory|\nomefile.mdf”).

Perché si possa continuare ad usare DataDirectory è necessario farlo puntare alla cartella contenente il database.

Un approccio intuitivo è quello di aggiungere la notazione “..\..\” al percorso del file, per farlo puntare alla cartella del progetto:

|DataDirectory|**\..\..**nomefile.mdf

Purtroppo questa possibilità non è ammessa.

La soluzione è quella di impostare da codice la cartella associata a DataDirectory.

```
string baseDir =  Path.Combine(AppDomain.CurrentDomain.BaseDirectory, @"..\..\"); ;
BaseDir = Path.GetFullPath(baseDir);    // necessario!
AppDomain.CurrentDomain.SetData("DataDirectory",baseDir);
```

Si combina la cartella di output con il percorso relativo “..\..\”, ottenendo il percorso della cartella del progetto. Si trasforma questo percorso nella notazione assoluta e quindi si usa per impostare la nuova cartella associata a DataDirectory.

1.6 Uso di un database Access

L'uso di database Access (estensioni ".accdb" e ".mdb") non implica particolari differenze rispetto a quanto visto finora.

Aggiungendo il database al progetto, questo viene gestito da VS, e cioè:

1. Viene automaticamente creata una connessione su Server Explorer.
2. Viene copiato sulla cartella di output durante la compilazione.

1.6.1 Stringa di connessione database Access

Vale quanto detto in precedenza sui percorso assoluti e relativi. Se si desidera accedere al database mediante un percorso relativo si può utilizzare la notazione |DataDirectory|

```
Provider=Microsoft.ACE.OLEDB.12.0;
Data Source=|DataDirectory|\Biblioteca.accdb;Persist Security Info=False";
```

La memorizzazione della stringa di connessione in App.Config segue le stesse regoleviste con LocalDB:

```
<connectionStrings>
    <add name="DemoConsole.Biblioteca"
         connectionString="Provider=Microsoft.ACE.OLEDB.12.0;
                           Data Source=|DataDirectory|\Biblioteca.accdb;
                           Persist Security Info=False"
         providerName="System.Data.OleDb" />
</connectionStrings>
```

Provider "Microsoft.ACE.OLEDB.12.0"

L'accesso ai database Access richiede che sul computer sia installato il provider "**Microsoft.ACE.OLEDB.12.0**". Il provider è installato automaticamente con Access 2007+. Alternativamente, può essere installato direttamente con il file "AccessDatabaseEngine.exe". (nella versione 32/64 bit, in base al SO intallato).

Problemi di accesso a database Access.

Successivamente all'installazione di Access 2013 e VS2013 ho riscontrato l'impossibilità di accedere da programma ai database Access (creati con Access 2003). Ho risolto installando la versione 2007 di AccessDatabaseEngine.exe (lingua inglese).

Occorre verificare sul proprio sistema se vi sono problemi con database Access all'interno di programmi .NET. In caso positivo, consiglio di provare a installare il file AccessDatabaseEngine_X64_2007_EN.exe", collocato in pubblica.

1.6.2 Copia del database nella cartella di output

Diversamente da quando accade con SQL Server, la copia ripetuta di un database Access nella cartella di output non produce alcun problema. Non è quindi necessario, (benché resti possibile) impostare da codice una cartella diversa per DataDirectory.