Trysil: Un ORM "light" per Delphi

David Lastrucci
Open Source Italia S.r.l.

DelphiDay italian conference ONLINE EDITION

David Lastrucci

Ho iniziato a Sviluppare in Delphi dalla versione 1

Lavoro in Open Source Italia dove sono amministratore e R&D Manager

Seguo i componenti ed i wizard che vengono utilizzati per lo sviluppo del Software Gestionale OS1

Seguo e coordino il team di Ricerca & Sviluppo nella realizzazione del nuovo prodotto in cloud di OSItalia

david.lastrucci@ositalia.com
david.lastrucci@gmail.com
Skype: david.lastrucci

https://ositalia.com
https://github.com/davidlastrucci

DelphiDay italian conference ONLINE EDITION

Agenda

- Introduzione
- Database supportati, SqlSyntax e Database Drivers
- TTConnection
- Entity e Attributes
- TTContext, TTMetadata e TTIdentityMap
- TTLazy<T> e TTLazyList<T>
- Utilizzo senza Lazy
- TTSession
- TTEvent e TTEvent<T>

ORM

Un ORM (Object-Relational Mapping) è una libreria software che favorisce l'integrazione tra due sistemi che apparentemente potrebbero sembrare incompatibili:

- Programmazione a oggetti (OOP)
- Database relazionali (RDBMS)

Dietro a comandi piuttosto semplici, un ORM, nasconde le attività di lettura e manipolazione dei dati che, utilizzando il linguaggio SQL, risulterebbero complesse e, molto spesso, ripetitive

Trysil

Trysil è un ORM "light" per Embarcadero Delphi

Permette di approcciare, in modalità Object Oriented, i servizi relativi alla persistenza dei dati su alcuni database relazionali

E' un progetto Open Source disponibile su GitHub

https://github.com/davidlastrucci/trysil

Perché Trysil?

Durante la seconda guerra mondiale ci fu un'operazione britannica che aveva lo scopo di costituire una base di accoglienza a Trysil

Trysil, oggi una importante località per sport invernali, si trova nella parte orientale della Norvegia e, a quei tempi, era occupata dai tedeschi

L'operazione britannica venne chiamata ORM!

http://codenames.info/operation/orm/

Database supportati

Supporta, ad oggi, i seguenti database:

- FirebirdSQL
- SQLite*
- Microsoft SQL Server (dalla versione 2012*)

* Fa uso dei Sequence per le chiavi primarie; il driver per SQLite, al posto dei Sequence, utilizza la colonna ROWID (MAX+1)

SqlSyntax

Le sintassi dei database le troviamo nella cartella Data\SqlSyntax, all'interno delle seguenti unit:

- Trysil.Data.SqlSyntax.pas
- Trysil.Data.SqlSyntax.FirebirdSQL.pas
- Trysil.Data.SqlSyntax.SQLite.pas
- Trysil.Data.SqlSyntax.SqlServer.pas

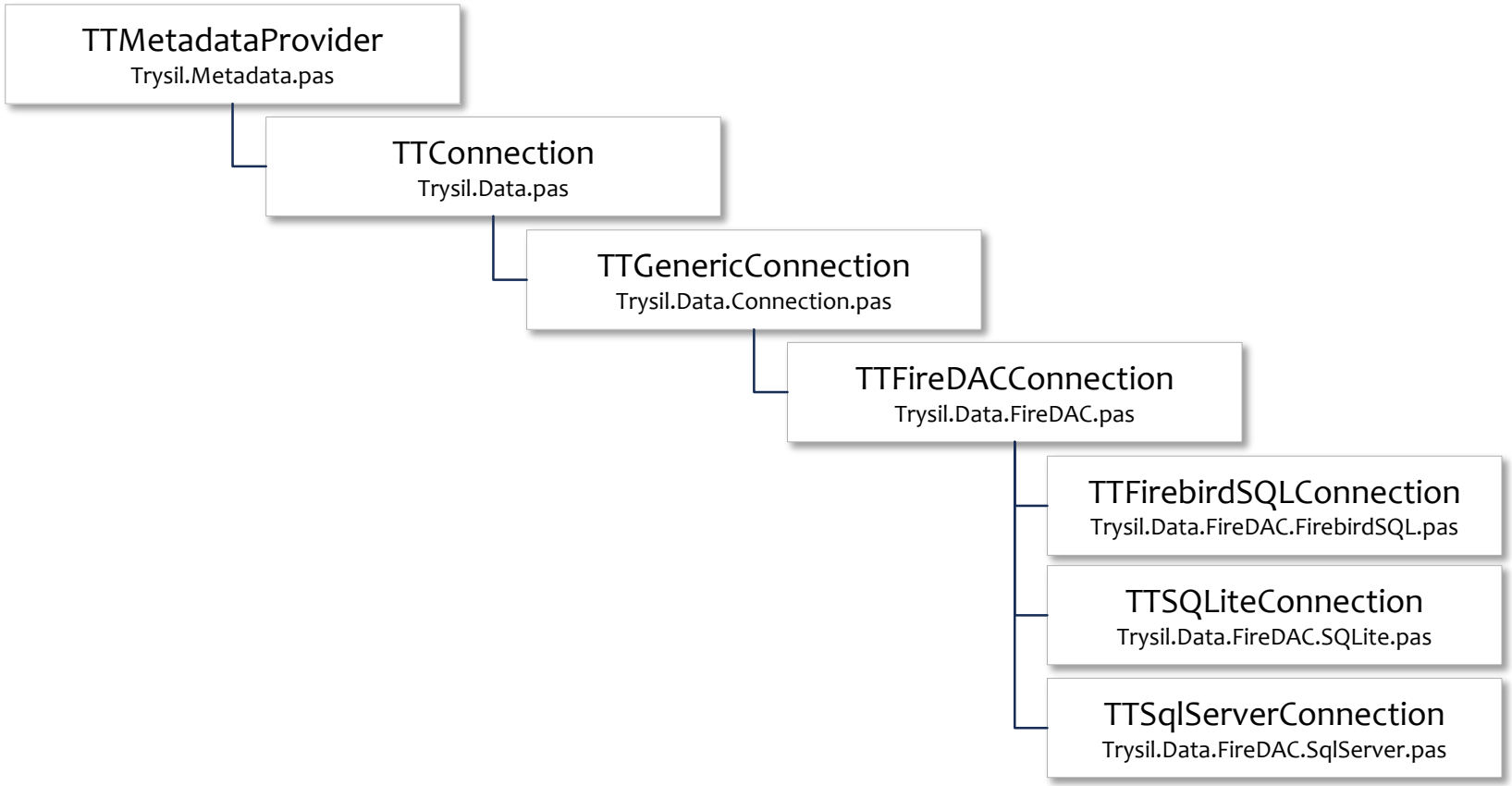
Database drivers

I driver dei database gli troviamo all'interno delle seguenti unit:

- Trysil.Data.pas
- Trysil.Data.Connection.pas
- Trysil.Data.FireDAC.pas
- Trysil.Data.FireDAC.FirebirdSQL.pas
- Trysil.Data.FireDAC.SQLite.pas
- Trysil.Data.FireDAC.SqlServer.pas

TTConnection

TTConnection è la connessione astratta ad un database



Entity

Le Entity sono classi

Non necessariamente devono ereditare da una classe padre

Unico vincolo è che devono avere un costruttore con un unico parametro di tipo TTContext oppure il costruttore di default (quello senza parametri)

Attributes

Gli attributi vengono utilizzati per decorare le Entity

Sono disponibili i seguenti:

Per le Entity

- TTableAttribute
- TSequenceAttribute
- TRelationAttribute

Per i Field delle Entity

- TPrimaryKeyAttribute
- TColumnAttribute
- TDetailColumnAttribute
- TVersionColumnAttribute

TTContext

TTContext è la classe che si usa per eseguire le operazioni CRUD sul database

```
procedure SelectAll<T>(const AResult: TTList<T>);
procedure Select<T>(const AResult: TTList<T>; const AFilter: TTFilter);
function Get<T>(const AID: TTPrimaryKey): T;

procedure Insert<T>(const AEntity: T);
procedure Update<T>(const AEntity: T);
procedure Delete<T>(const AEntity: T);
```

TTMetadata

Le operazioni di Insert, Update e Delete sul database utilizzano i parametri

Per questo motivo abbiamo bisogno dei metadati

I metadati di una Entity contengono:

- TableName
- PrimaryKey
- Columns
  - ColumnName
  - DataType
  - DataSize

TTIdentityMap

TTEntityMap è una cache delle Entity in memoria affidata a TTContext

Se la Entity richiesta è già stata letta dal database, tramite l'utilizzo di TTIdentityMap, viene restituita la stessa istanza della Entity letta precedentemente

L'utilizzo di TTIdentityMap garantisce comunque dati aggiornati

TTIdentityMap può essere disabilitato durante la creazione di TTContext

TTLazy<T> e TTLazyList<T>

TTLazy<T> e TTLazyList<T> consentono la lettura dal database solo quando abbiamo davvero bisogno di avere i dati (lazy load, caricamento pigro)

TTLazy<T> è utilizzata per le "Foreign Entity" (Cliente.Nazione)

TTLazyList<T> è utilizzata per le "Detail Entity" (Fattura.Righe)

Utilizzo senza Lazy

E' possibile utilizzare Trysil anche senza i field di tipo TTLazy<T> e TTLazyList<T>

Possiamo usare la classe T al posto di TTLazy<T> e TList<T> oppure TObjectList<T> (di System.Generics.Collections) al posto di TTLazyList<T>

Trysil però non è in grado di gestire questi oggetti che quindi devono essere creati e distrutti manualmente

No TTIdentityMap

TTSession<T>

TTSession<T> permette fare Insert, Update e Delete di Entity dello stesso tipo T senza aggiornare il database

L'elenco delle operazioni effettuate durante una TTSession<T> vengono applicate solamente quando viene richiamato il metodo ApplyChanges

Se il database le supporta, tutte le operazioni vengono applicate in transazione

```
function TTContext.CreateSession<T>(const AList: TList<T>): TTSession<T>;
```

TTEvent e TTEvent<T>

Alle Entity, tramite attributi specifici, possono essere associati eventi

- TInsertEventAttribute
- TUpdateEventAttribute
- TDeleteEventAttribute

TTEvent e TTEvent<T>

Gli attributi specificano una classe di tipo TTEvent che al suo interno definisce i metodi DoBefore e DoAfter

All'interno dell'evento sono disponibili il TTContext e la Entity sulla quale stiamo facendo un'operazione di Insert, Update o Delete

Durante gli eventi BeforeUpdate e BeforeDelete abbiamo anche la possibilità di accedere alla OldEntity

Grazie per l'attenzione