Projektowanie aplikacji bazodanowych w .NET

Paweł Rajba

ADO.NET Entity Framework

- Plan wykładu
  - Wprowadzenie
  - Architektura i tworzenie modelu
  - Przegląd możliwości
    - Asocjacje, strategie dziedziczenia
    - Komponenty, klucze główne
    - Wzorce active record i unit of work
    - Transakcje
    - Miękkie usuwanie obiektów
    - Optymistyczna współbieżność
    - Pobieranie danych
  - Testy wydajnościowe

Wprowadzenie

- Co to jest ADO.NET Entity Framework
  - ORM
  - Krótkie odniesienie do LINQ To SQL
    - Brak innego języka niż LINQ
    - Nie ma relacji M:N
    - Dziedziczenie: tylko tabela na hierarchię
    - Pojedyncza tabela z wielu encji: nie ma
    - Ścisły związek ze schematem bazy danych
  - Przewaga EF nad XPO lub Castle Active Record: LINQ
- Jak zacząć
  - Visual Studio 2008 SP1
  - Ładny ,,designer'' w Visual Studio (demo później)

Wprowadzenie

- Trochę literatury na początek
  - Dokument przeglądowy o EF
    - http://msdn.microsoft.com/en-us/library/aa697427(VS.80).aspx
  - Wprowadzenie do EF
    - http://msdn.microsoft.com/en-us/library/bb399567.aspx
  - Blog twórców EF:
    - http://blogs.msdn.com/adonet/archive/tags/Entity+Framework/default.aspx
  - Przykłady mapowań:
    - http://download.microsoft.com/download/b/3/3/b333d63e-0df2-4d43-978a-1ce9d2f39801/EntityFrameworkMappingWhitepaper.pdf
  - Przegląd możliwości:
    - http://msdn.microsoft.com/en-us/library/bb738510.aspx
  - Ciekawe podsumowanie:
    - http://www.cnblogs.com/levinknight/archive/2008/06/11/1217965.html

Wprowadzenie

- Wsparcie dla innych platform
  - SQL Server jest wspierany przez same VS2008 po instalacji SP1
  - Oracle, PostgreSQL, MySQL jest wspierany przez produkt dotConnect firmy DevArt: http://devart.com/dotconnect/ (rozwiązanie pełne jest płatne
    - cennik dla Oracle: http://devart.com/dotconnect/oracle/ordering.html (ceny od $150 za licencję do $1800 dla całej firmy)
  - Inne rozwiązanie dla Oracle (EF Oracle Sample Provider): http://code.msdn.microsoft.com/EFOracleProvider/Release/ProjectReleases.aspx?ReleaseId=1395 http://blogs.msdn.com/jkowalski/archive/2008/06/23/sample-entity-framework-provider-for-oracle.aspx

Architektura modelu

- Literatura
  - Dokument z opisem
    - http://msdn.microsoft.com/en-us/library/aa697428(VS.80).aspx
  - Krótki przegląd
    - http://zeetalks.wordpress.com/2008/11/26/vs-2008-entity-framework-entities-and-relationships-in-the-entity-data-model/
- Składowe modelu
  - schemat konceptualny
    - opisywany przez CSDL (conceptual schema definition language)
  - schemat logiczny bazy danych
    - opisywany przez SSDL (store schema definition language)
  - mapowanie CS na SS
    - opisywane przez MSL (mapping specification language)
- Wszystkie CSDL, SSDL i MSL mają składnię XML-ową

Architektura modelu

- Składowe są zawarte w plikach XML o rozszerzeniach: .csdl, .ssdl i .msl
  - pliki musza być oznaczone jako embedded resource
  - ,,Designer" VS umieszcza zawartość plików w jednym pliku .edmx
- W pliku App.config powinien być connectionstring Przykładowy:

```
<add  name="BibliotekaEntities„
        connectionString=„
                metadata=.\Model.csdl|.\Model.ssdl|.\Model.msl;
                provider=System.Data.SqlClient;
                provider connection string=&quot;
                        Data Source=.;
                        Initial Catalog=KursPBD;Integrated Security=True;
                        MultipleActiveResultSets=True&quot;„
        providerName="System.Data.EntityClient" />
```

Architektura modelu

- Terminologia związana z Entity Framework
  - TYPY
    - EntityType –typ encji
    - Property – właściwość encji
      - może być typu SimpleType, ComplexType, RowType
    - EntityKey – zbiór Propertys jednoznacznie określający encję
    - ComplexType – zbiór propertys typu SimpleType, ComplexType, RowType
      - od encji różni się tym, że nie ma klucza, tzn. EntityKey
    - RowType – analogiczny do ComplexType, ale może brać udziału w dziedziczeniu
    - RelationshipType – reprezentuje powiązania między encjami (Association i Containment)
  - INSTANCJE
    - EntitySet – zbiór instancji obiektów typu EntityType (lub ich podtypów)
    - RelationshipSet – reprezentuje zbiór powiązań pomiędzy zbiorami encji (EntitySet)
- Specyfikacja do opisu plików CSDL, SSDL, MSL: http://msdn.microsoft.com/en-us/library/aa697428(VS.80).aspx

Architektura modelu

- Atrybuty do opisu klas encji
  - EdmComplexProperty – określa mapowanie na właściwość ComplexType
  - EdmComplexType – określa typ (klasę) jako odpowiadającej ComplexType
  - EdmEntityType – określa, że dana klasa mapuje się na encję
  - EdmScalarProperty – określa skalarną właściwość
  - EdmRelationshipNavigationProperty – określa, że dana właściwość jest elementem relacji
  - EdmRelationship – określa relację pomiędzy encjami
    - definiowany na poziomie assembly
- Specyfikacja atrybutów do opisu klas: http://msdn.microsoft.com/en-us/library/bb738639.aspx

Architektura modelu

- Podsumowując, utworzenie modelu wymaga
  - wprowadzenie do pliku App.config odpowiedniego connectionstring
  - utworzenie plików CSDL, SSDL, MSL
  - utworzenie klas modelu opatrzonych odpowiednimi atrybutami
  - opatrzenie assembly z modelem atrybutem EdmSchema

Architektura modelu

- Stany obiektów
  - Added
  - Deleted
  - Modified
  - Unchanged
  - Detached

Przykład

- SimpleMapping

Tworzenie modelu

- Są trzy główne scenariusze budowania struktury:
  - dany jest schemat bazy danych i na jego podstawie tworzony jest model obiektowy
    - wspierany przez VS, można też skorzystać z narzędzi do generowania kodu (np. CodeSmith)
  - dany jest model obiektowy, na podstawie którego tworzony jest schemat bazy danych
    - nie znalazłem żadnego wsparcia
  - dany jest schemat w np. UML i na jego podstawie generowane są wszystkie składowe
    - być może warto sprawdzić, czy np. Sybase oferuje wsparcie

Tworzenie modelu

- Mamy kilka sposobów utworzenia schematu.
  - Skorzystanie z ,,designera'' wbudowanego w VS
  - Skorzystanie z narzędzia edmgen.exe
    - nie testowałem
  - Samodzielne utworzenie plików
    - tutaj bardzo pomocne są narzędzia typu CodeSmith
- Tworzenie modelu obiektowego za pomocą ,,designera'' VS
- Przykładowe utworzenie modelu:
  - Najpierw tworzymy schemat w bazie danych (odpalamy skrypt Biblioteka.sql)
  - Tworzymy nowy projekt VS, dodajemy ADO.NET Entity Data Model, postępujemy zgodnie z instrukcjami

Tworzenie modelu

- Kilka linków
  - Kompletny przykład: http://msdn.microsoft.com/en-us/library/bb399790.aspx
  - Przykład manualnie stworzonego modelu:
    - konfiguracja XML: http://msdn.microsoft.com/en-us/library/bb399785.aspx
    - pliki klas: http://msdn.microsoft.com/en-us/library/bb738583.aspx

Przegląd możliwości

- Czego oczekujemy od systemu ORM?
  - Automatycznego utworzenia/aktualizacji schematu bazy danych (nie ma)
  - Asocjacji, kaskadowości operacji, leniwego dociąganie, nawigacji
  - Strategii dziedziczenia
  - Realizacji komponentów (nested data)
  - Autogenerowania kluczy głównych, złożonych kluczy głównych
  - Wzorca ,,Active Record''
  - Wzorca ,,Unit of Work''
  - ,,Miękkiego'' usuwania obiektów
  - Transakcji, optymistycznej współbieżności
  - Pobierania wybranego fragmentu danych (stronicowanie)
  - Leniwego dociąganie pól typów skalarnych (nie ma)

Przegląd możliwości: asocjacje

- EF wspiera wszystkie rodzaje asocjacji: 1-1, 1-*, *-*
- Zapis odbywa się kaskadowo
  - jest to dosyć sprawne – wystarczy jakkolwiek wskazać asocjację, a zostanie ona zapisana
- Relacja jeden-do-wielu
  - Domyślnie przy usuwaniu parenta: parent jest usuwany, a odpowiadających childach jest ustawiany null
  - Usuwanie kaskadowe parent -> child można ustawić wprowadzając w pliku CSDL taki wpis:

```
<End Role="Ksiazka" Type="Model.Parent" Multiplicity="0..1">
    <OnDelete Action="Cascade" />
</End>
```

Przegląd możliwości: asocjacje

- Relacja wiele-do-wielu
  - Usunięcie obiektu po którejkolwiek stronie implikuje również usunięcie powiązanych asocjacji w tabeli łączącej
  - Nie ma kaskadowości
    - dokładniej: nie można zdefiniować ondelete=cascade, jeśli koniec asocjacji ma mnogość *
  - Uwaga: jeśli w bazie danych mamy relacje: Left, Right i LeftRight, przy czym LeftRight ma dokładnie dwa pola będące kluczami obcymi do tabel Left i Right, Designer VS rozpozna automatycznie relację wiele-do-wielu.
- Uwaga: nie ma w ogóle mowy o kaskadowości w odniesieniu do operacji UPDATE

Przegląd możliwości: asocjacje

- Ładowanie referencji:
  - domyślnie obiekty referencji nie są ładowane
  - żeby mieć dostęp do wskazywanego obiektu/kolekcji trzeba wywołać metodę Load()
  - aby sprawdzić czy obiekt/kolekcja jest załadowana mamy właściwość IsLoaded
  - konstrukcja dla kolekcji (w przypadku automatycznie wygenerowanego modelu):
    - context.Parent.First().K_Child.Load();
  - konstrukcja dla obiektu (w przypadku automatycznie wygenerowanego modelu):
    - context.Child.First().ParentReference.Load();

Przegląd możliwości: asocjacje

- Podsumowując:
  - Dostępne jest tylko leniwe ściąganie asocjacji i gorliwe pobieranie właściwości skalarnych.
  - Gorliwość pobierania asocjacji można rozwiązać odpowiednio oprogramowując get w klasach modelu.
  - FetchMode.Join (z Hibernate) można zasymulować zadając zapytanie typu outer join w Linq

Przykłady

- AssociationOne2Many
- AssociationMany2Many
- AutoGenerated

Przegląd możliwości: dziedziczenie

- Mamy trzy główne strategie dziedziczenia
  - tabela na hierarchię klas
  - tabela na podklasę (joined subclass)
  - tabela na klasę
- Artykuł opisujący jak zrobić tabelę na hierarchię klas: http://mosesofegypt.net/post/Inheritance-and-Associations-with-Entity-Framework-Part-1.aspx
- Filmik pokazujący zrealizować tabelę na podklasę: http://msdn.microsoft.com/en-us/data/cc765425.aspx

Przykłady

- Inheritance, InheritanceGenerated
  - Strategia joined subclass
- Inheritance2, Inheritance2Generated
  - Strategia tabela na hierarchię
- Inheritance3
  - Strategia joined subclass (hierarchia z ,,rozwidleniem'')
  - Warto zwrócić uwagę, że do pobrania danych po stronie SQL wykorzystywany jest UNION ALL

Przegląd możliwości: komponenty

- Możemy zrealizować dwa scenariusze
  - Vertical Entity Splitting (inaczej komponenty lub ,,nested data'')
    - Klasę
      Osoba(Id, Imie, Nazwisko, Ulica, Miasto)
      można składować w dwóch tabelach
      Osoba(Id, Imie, Nazwisko) i Adres(Id, Ulica, Miasto)
  - Horizontal Entity Splitting (bonus)
    - Klasę
      Pracownik(ID, Imie, Nazwisko, bool Kierownik)
      można składować w dwóch tabelach
      Pracownik(ID, Imie, Nazwisko) i Kierownik(ID, Imie, Nazwisko)
      (wybór tabeli jest na podstawie pola Kierownik)

Przykłady

- NestedData
- HorizontalSplitting

Przegląd możliwości: klucze główne

- Opis nt. kluczy w EF: http://msdn.microsoft.com/en-us/library/dd283139.aspx
- Aby uruchomić automatyczne generowanie kluczy należy:
  - Tabele w BD utworzyć dodając identity w definicji kolumny klucza
  - Dodać odpowiednią informację w pliku SSDL
    <Property Name="ID" Type="int" Nullable="false" StoreGeneratedPattern="Identity" />
  - Uwaga: można wprost nadać wartość polu ID w obiekcie, ale zostanie ona zignorowana
- A jak to jest w Oracle?
  - Korzystając ze sterownika firmy Devart mamy następujące rozwiązanie:
    - po stronie aplikacji ustawiamy model analogicznie jak w przypadku MS SQL
    - po stronie bazy danych tworzymy sekwencję i odpowiedni wyzwalacz
  - Aktualnie nie ma wsparcia do generowania ID po stronie sterownika i aplikacji.
- Jest wsparcie dla kluczy złożonych

Przykłady

- KeysAutogenerated
- CompositeKeys

Przegląd możliwości: wzorce active record i unit of work

- Wzorzec ActiveRecord
  - Trzeba sobie zaimplementować ręcznie
  - Przykładowa implementacja: ActiveRecordPattern
- Wzorzec UnitOfWork
  - Jest domyślnym sposobem nanoszenia zmian

Przegląd możliwości: „miękkie” usuwanie obiektów

- Cel jest taki, żeby usunięcie obiektu sprowadzało się do zmiany wybranego pola rekordu, a nie do fizycznego usunięcia tego rekordu
  - Nie ma wsparcia w Entity Framework
- Rozwiązanie problemu
  - Dodanie dodatkowego pola w schemacie (tak jak w XPO)
    - kłopot w tym rozwiązaniu jest taki, że przy pobieraniu trzeba w klauzuli WHERE dodawać odpowiedni warunek odfiltrowujący usunięte wiersze
  - Dla każdej tabeli (encji) utworzyć dualną z usuniętymi rekordami – usunięcie polegałoby na przeniesieniu rekordu z tabeli właściwej do archiwalnej.
    - kłopot w tym, że liczba tabel jest podwojona

Przykład

- DeleteWithMarking

Przegląd możliwości: transakcje

- Można je realizować korzystając z
  - System.Transactions.TransactionScope

Przykład

- Transakcje

Przegląd możliwości: optymistyczna współbieżność

- Włączenie dla pola w obiekcie optymistycznej współbieżności sprowadza się do ustawienia
  - w designerze VS właściwości Concurrency Mode na Fixed, co w schemacie CSDL przekłada się na wpis ConcurrencyMode="Fixed"
  - Przykładowa właściwość:
    - <Property Name="Nazwisko" Type="String" MaxLength="200" Unicode="false" FixedLength="false" ConcurrencyMode="Fixed" />

Przegląd możliwości: optymistyczna współbieżność

- Jak to działa?
  - Framework zapamiętuje wartość oryginalną z bazy danych
  - W momencie zapisu zmienionego obiektu w warunku where zapytania dołączane są poprzednie wartości i w przypadku ingerencji w dane przez osoby trzecie, liczba zmodyfikowanych rekordów to 0, to generuje wyjątek po stronie aplikacji
  - Przykładowe zapytanie generowane przez EF:
    exec sp_executesql N'update [dbo].[Osoba]
    set [Nazwisko] = @0
    where ((([ID] = @1) and ([Imie] = @2)) and ([Nazwisko] = @3))
    ',N'@0 varchar(13),@1 int,@2 varchar(4),@3
    varchar(8)',@0='Nowe nazwisko',@1=1,@2='Imie',@3='Nazwisko‘
- Dodatkowe rozważania w przypadku obiektów odłączonych:
  - http://blogs.msdn.com/cesardelatorre/archive/2008/09/05/optimistic-concurrency-updates-using-entity-framework-in-n-tier-and-n-layer-applications-part-2.aspx

Przykład

- OptimisticConcurrency

Przegląd możliwości: pobieranie danych

- Języki
  - LINQ to Entities
  - EntitySQL
    - Dokumentacja pod adresem: http://msdn.microsoft.com/en-us/library/bb399560.aspx
- Stronicowanie
  - Jest dostępne poprzez kombinacje operatorów LINQ Skip i Take

Testy wydajnościowe

- Porównanie zostało wykonane na SQL Server 2005 dla Entity Framework, Devexpress XPO, Castle ActiveRecord (NHibernate)
- Wyniki
  - Tworzenie obiektów:
    - EF: ok. 36s.
    - XPO: ok. 65s.
    - XPO (UoW): ok. 15s.
    - Castle AR: ok. 93s.
  - Wyciąganie obiektów (proste złączenie + warunek)
    - EF: ok. 3,1s
    - XPO: ok. 1,2s (prostsze zapytanie – LINQ to XPO nie obsługuje złączeń)
    - Castle AR: ok. 4,4s
  - Aktualizacje wybranych obiektów
    - EF: 3,4s. (UoW, bez UoW)
    - XPO: ok. 7,6
    - XPO: ok. 2,5 (UoW)
    - Castle AR: ok. 8s.