Projektowanie aplikacji bazodanowych w .NET

Wykład 7

Paweł Rajba

Instytut Informatyki

Uniwersytet Wrocławski

Plan wykładu

- LINQ to SQL
  - Krótki przegląd
  - Tworzenie modelu
    - automatycznie
    - manualnie
  - Optymistyczna współbieżność
  - Dziedziczenie
  - *Lazy loading*
  - Narzędzia

Krótki przegląd

- Umożliwia mapowanie ORM
- Obsługuje ważniejsze aspekty jak asocjacje czy dziedziczenie
- Jest rozwiązaniem lekkim: czyli działa dosyć szybko, ale mało funkcjonalności
- Pozwala na zadawanie zapytań w LINQ
  - nie pozwala zadawać zapytań w języku literalnym
- Strona: *http://msdn.microsoft.com/en-us/library/bb386976.aspx*

Tworzenie modelu

- Wsparcie obejmuje tworzenie klas na podstawie bazy danych
- Klasy można utworzyć
  - Z automatu wbudowanego w VS2008
    - tutaj problemem jest nazewnictwo oraz brak implementacji wzorca ActiveRecord
  - Za pomocą programu sqlmetal
    - narzędzie wykonywane z wiersza poleceń
    - możliwości podobne do automatu VS
  - Manualnie oraz poprzez samodzielne utworzenie klas i dodanie odpowiednich atrybutów
    - W połączeniu z np. CodeSmith wydaje się sensowniejszym rozwiązaniem

Tworzenie klas: sqlmetal

- Program wykonywany z wiersza poleceń
- Domyślna lokalizacja:

  C:\Program Files\Microsoft SDKs\Windows\vn.nn\bin

  - U mnie:

    C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\sqlmetal.exe
- Wynikiem może być:
  - plik z klasami wraz z mapowaniem
  - osobno plik z klasami i plik z mapowaniem

    (ewentualnie samo mapowanie)
  - plik dbml (plik schematu)

Tworzenie klas: sqlmetal

- Do działania potrzebujemy:
  - Podpiąć do projektu plik z klasami (.cs)
  - W kodzie załadować plik z mapowaniem (opcjonalne)
- Przykładowe użycia:
  - `sqlmetal.exe /code:C:\northwind.cs /server:. /database:northwind`
  - `sqlmetal.exe /code:C:\northwind.cs /map:C:\northwind.xml /server:. /database:northwind`
- Strona programu sqlmetal:

  http://msdn.microsoft.com/en-us/library/bb386987.aspx

  Koncepcja generowania kodu:

  http://msdn.microsoft.com/en-us/library/bb399400.aspx

Tworzenie klas: automat VS

- Jak korzystamy?
  - Zaznaczamy projekt, prawy przycisk myszy, wybieramy *Add* i *New Item*
  - Z listy wybieramy *LINQ to SQL Classes*
  - Otwieramy widok *Server Explorer*, wybieramy połączenie (ewentualnie najpierw je tworzymy), rozwijamy drzewo i po prostu przeciągamy tabelki na forma
- Co powstaje?
  - plik .dbml i plik .cs
  - dodatkowo jest plik pomocniczy z układem designera

Tworzenie klas: sqlmetal i VS

Podstawowe konstrukcje dla utworzonych klas

- Modyfikacja:

```
var dc = new DataClassesDataContext( "connectionstring" );
var result = from o in dc.Osobas select o;
foreach ( var os in result ) os.Nazwisko = "inne";
dc.SubmitChanges();
```

- Dodanie obiektu:

```
var dc = new DataClassesDataContext( "connectionstring" );
dc.Osobas.InsertOnSubmit(
    new Osoba { ID = 100, Imie = "J", Nazwisko = "K" } );
dc.SubmitChanges();
```

- Usunięcie obiektu:

```
var dc = new DataClassesDataContext( "connectionstring" );
var result = from o in dc.Osobas select o;
dc. DeleteAllOnSubmit( result );
dc.SubmitChanges();
```

Tworzenie klas manualnie

- Tworzymy najpierw zwykłą klasę (POCO)
- Następnie dodajemy odpowiednie atrybuty:
  - Table
  - Column
  - Association
  - InheritanceMapping
  - (...)

  Atrybuty są opisane pod tym adresem:

  http://msdn.microsoft.com/en-us/library/bb386971.aspx
- Można też utworzyć plik XML z mapowaniem

  Opis: *http://msdn.microsoft.com/en-us/library/bb386907.aspx*

Tworzenie klas manualnie

- Asocjacje tworzymy atrybutem *Association* oraz poprzez klasy *EntityRef* i *EntitySet*
  - Opis: http://msdn.microsoft.com/en-us/library/bb386950.aspx
- Dostępne są relacje jeden-do-jeden, jeden-do-wielu
  - relacje w-do-w trzeba symulować przez relacje j-do-w
- Schemat klasy *Parent'a* dla podstawowej relacji jeden-do-wielu

```
class Parent {
    private EntitySet<Child> ChildrenStorage;
    ...
    [Association( Storage="ChildrenStorage", OtherKey="Parent_ID" )]
    public EntitySet<Child> K_Child {
        get { return this.ChildrenStorage; }
        set { this.ChildrenStorage.Assign( value ); }
    }
}
```

Tworzenie klas manualnie

- Schemat klasy *Child'a* dla podstawowej relacji jeden-do-wielu

```
class Child {
    private EntityRef<Parent> ParentStorage;
    ...
    [Column]
    private int? Parent_ID { get; set; }
    [Association(
        Storage="ParentStorage",
        ThisKey="Parent_ID",
        IsForeignKey=true )]
    public Parent Parent {
        get { return this.ParentStorage.Entity; }
        set { this.ParentStorage.Entity = value; }
    }
}
```

Optymistyczna współbieżność

- Realizowana poprzez wyjątek *System.Data.Linq.ChangeConflictException*
- Konflikty są dostępne w kolekcji DataContext.ChangeConflicts
- Każdy z nich ma metodę Resolve, dzięki której możemy określić sposób rozwiązania konfliktu.
- Mamy trzy wartości:
  - A: oryginalną z bazy danych (w momencie pobrania danych przez program)
  - B: zmienioną w międzyczasie przez inny proces (już po pobraniu przez program)
  - C: nową wprowadzoną przez program

Optymistyczna współbieżność

- To, która z nich zostanie ostatecznie w bazie danych określają stałe przekazywane jako parametr metody *Resolve*:
  - C: *RefreshMode.KeepChanges*
  - B: *RefreshMode.OverwriteCurrentValues*
  - A: *RefreshMode.KeepCurrentValues*

Przykłady

- `SimpleExample`
- `AssociationOneToOne`
- `AssociationOneToMany`

Dziedziczenie

- Dostępna jest strategia *tabeli na hierarchię klas*
- Przypuśćmy, że mamy klasy *Osoba* $\rightarrow$ *Pracownik*

  W celu implementacji dziedziczenia należy:
  - Dodać atrybuty do klasy bazowej wskazując klasę domyślną:

```
[Table]
[InheritanceMapping(Code = 1, Type = typeof(Osoba), IsDefault=true)]
[InheritanceMapping(Code = 2, Type = typeof(Pracownik))]
public class Osoba { ... }
```

  - W klasie bazowej dodać właściwość dykryminatora:

```
[Column(IsDiscriminator=true)]
public int TypOsoby { get; set; }
```

Ciekawy artykuł: *http://blogs.microsoft.co.il/blogs/bursteg/archive/2007/10/01/linq-to-sql-inheritance.aspx*

Przykład

- `Inheritance`

Lazy loading

- Domyślnie wszystko jest dociągane
- Można wyłaczyć dociąganie brakujących obiektów:

  $ctx.DeferredLoadingEnabled = false$
- Można to zmienić przez ustawienie opcji w właściwości context.LoadOptions
  - LoadWith określa co ma być ściągane z czym, np. pobranie osoby implikuje pobranie jednostki (robiony jest JOIN)
  - Associate pozwala określić dociąganie warunkowe (również robiony jest JOIN)

Lazy loading

- Linki:
  - DataLoadOptions:

    http://msdn.microsoft.com/en-us/library/system.data.linq.dataloadoptions.aspx
  - LoadWith:

    http://msdn.microsoft.com/en-us/library/bb548760.aspx
  - AssociateWith:

    http://msdn.microsoft.com/en-us/library/bb534221.aspx

Przykład

- `LazyLoading`

Narzędzia

- Visual LINQ Query Builder

  http://code.msdn.microsoft.com/vlinq

  - Plugin do VS
- LINQPad

  http://www.linqpad.net/

Przykład

- Krótkie demo LINQPada