Сопоставление операций Entity Framework с хранимыми процедурами

Последнее обновление: 31.10.2015

Кроме определения отношений между сущностями Fluent API позволяет производить еще ряд действий, в частности, сопоставить стандартные операции Entity Framework, такие как добавление, обновление, удаление, с хранимыми процедурами. Подобная функциональность может быть нам необходимы, если мы хотим переопределить эти операции, либо если у нас уже есть база данных с хранимыми процедурами, и нам надо их использовать вместо стандартного механизма Entity Framework.

Допустим, у нас есть следующая модель Phone:

```
public class Phone
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Price { get; set; }
}
```

Теперь определим контекст данных:

```
public class MobileContext : DbContext
{
    public DbSet<Phone> Phones { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // переопределение добавления
        modelBuilder.Entity<Phone>()
            .MapToStoredProcedures(b => b.Insert(sp => sp.HasName("InsertPhone")
                .Parameter(pm => pm.Name, "Model")
                .Parameter(pm => pm.Price, "Price")
                .Result(rs => rs.Id, "Id")));

        // переопределение обновления
        modelBuilder.Entity<Phone>()
            .MapToStoredProcedures(b => b.Update(sp => sp.HasName("UpdatePhone")
                            .Parameter(pm => pm.Name, "Model")
                .Parameter(pm => pm.Price, "Price")
                .Parameter(pm => pm.Id, "Id")));

        // переопределение удаления
        modelBuilder.Entity<Phone>()
            .MapToStoredProcedures(b => b.Delete(sp => sp.HasName("DeletePhone")
                .Parameter(pm => pm.Id, "Id")));

        base.OnModelCreating(modelBuilder);
```

```
    }
}
```

С помощью метода MapToStoredProcedures осуществляется сопоставление операции с определенной сущностью с хранимыми процедурами. Так, первое блок осуществляет сопоставление операции добавления сущности Phone с хранимой процедурой InsertPhone:

```
modelBuilder.Entity<Phone>()
    .MapToStoredProcedures(b => b.Insert(sp => sp.HasName("InsertPhone")
        .Parameter(pm => pm.Name, "Model")
        .Parameter(pm => pm.Price, "Price")
        .Result(rs => rs.Id, "Id")));
```

Название процедуры задается с помощью метода `HasName()`. С помощью метода `Parameter()` происходит сопоставление определенных свойств модели Phone с параметрами процедуры. Так, в данном случае свойство Name сопоставляется с параметром model, а свойство Price с параметром price. И еще один метод `Result` позволяет указать свойство модели, которое будет принимать результат процедуры. В данном случае это свойство Id, которое получает идентификатор добавленного в бд объекта.

Пусть у нас уже есть база данных и таблица Phones, которая соответствует модели Phone и которая определяется следующим скриптом:

```
CREATE TABLE [dbo].[Phones] (
    [Id]    INT           IDENTITY (1, 1) NOT NULL,
    [Name]  NVARCHAR (50) NOT NULL,
    [Price] INT           NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

И теперь добавим к этой таблице хранимую процедуру (подробнее механизм добавления процедуры описан в статье Хранимые процедуры):

```
CREATE PROCEDURE [dbo].[InsertPhone]
        @model nvarchar(50),
        @price int
AS
        INSERT INTO Phones (Name, Price)
    VALUES (@model, @price * 66)

    SELECT SCOPE_IDENTITY() AS id
GO
```

Во-первых, название процедуры - InsertPhone - соответствует тому названию, которое используется выше в коде контекста. Во-вторых, процедуры имеет все те параметры, которые используются в коде контекста.

Для добавления используется выражение INSERT, а для возвращения результата выражение `SELECT SCOPE_IDENTITY() AS id`.

При этом добавление в бд в коде c# будет происходить как обычно:

```
using (MobileContext db = new MobileContext())
{
    Phone phone = new Phone { Name = "iPhone 6S", Price = 680 };
    db.Phones.Add(phone);
    db.SaveChanges();
    Console.WriteLine(phone.Id);
}
```

Только теперь Entity Framework будет для добавления использовать не собственное выражение, а код хранимой процедуры.

Подобным образом можно создать и использовать хранимую процедуру для обновления объекта:

```
CREATE PROCEDURE [dbo].[UpdatePhone]
        @model nvarchar(50),
        @price int,
        @id int
```

```
AS
        UPDATE Phones SET Name=@model, Price=@price WHERE Id=@id
GO
```

и для удаления объекта:

```
CREATE PROCEDURE [dbo].[DeletePhone]
        @id int
AS
        DELETE FROM Phones WHERE Id=@id
GO
```

Назад **Содержание** **Вперед**