Подход TPC

Последнее обновление: 31.10.2015

Подход TPC (Table Per Concrete Type / Таблица на каждый отдельный тип) предполагает создание для каждой модели по отдельной таблицы. Столбцы в каждой таблице создаются по всем свойствам, в том числе и унаследованным.

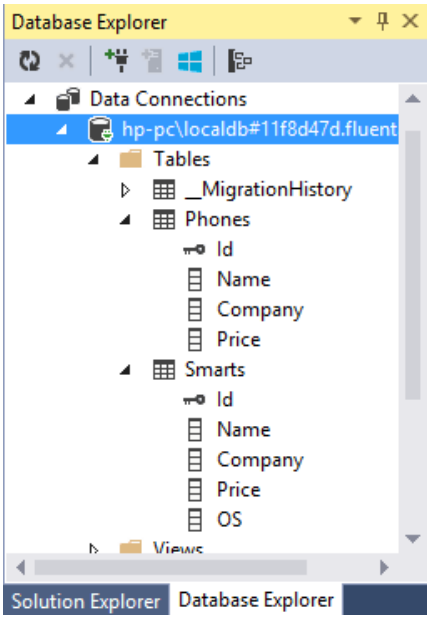
Чтобы применить подход, изменим объявления моделей и контекст следующим образом:

```
public class Phone
{
    [Key, DatabaseGenerated (DatabaseGeneratedOption.Identity)]
    public Guid Id { get; set; }
    public string Name { get; set; }
    public string Company { get; set; }
    public int Price { get; set; }
}

public class Smartphone : Phone
{
    public string OS { get; set; }
}

class MobileContext : DbContext
{
    public MobileContext() : base("DefaultConnection")
    { }
    public DbSet<Phone> Phones { get; set; }
    public DbSet<Smartphone> Smarts { get; set; }
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Phone>()
        .Map(m =>
        {
            m.MapInheritedProperties();
            m.ToTable("Phones");
        });
        modelBuilder.Entity<Smartphone>().Map(m =>
        {
            m.MapInheritedProperties();
            m.ToTable("Smarts");
        });
    }
}
```

Во-первых, обратите внимание, что у класса Phone в качестве типа ключа используется не int, а **Guid**. Это поможет нам избежать некоторых проблем с ключами. Хотя также можно было бы использовать `int` с ручной установкой Id при создании объекта.

Во-вторых, при настройке сопоставления моделей и таблиц у каждой модели вызывается метод **MapInheritedProperties()**, который указывает Entity Frameworky, что в таблицу для данной модели надо включить также наследуемые свойства, а не только те, которые определены непосредственно в этой модели.

При генерации базы данных у нас будут созданы две таблицы с полным набором столбцов:



Применение моделей:

```
using(MobileContext db = new MobileContext())
{
        db.Phones.Add(new Phone {Name = "Samsung Galaxy S5", Company = "Samsung", Price = 14000 });
        db.Phones.Add(new Phone {Name = "Nokia Lumia 630", Company = "Nokia", Price = 8000 });

        Smartphone s1 = new Smartphone { Name = "iPhone 6", Company = "Apple", Price = 32000, OS = "iOS" };
        db.Smarts.Add(s1);
        db.SaveChanges();

        foreach (Phone p in db.Phones)
                Console.WriteLine("{0} ({1}) - {2}", p.Name, p.Company, p.Price);
    Console.WriteLine();
    foreach (Smartphone p in db.Smarts)
        Console.WriteLine("{0} ({1}, {2}) - {3}", p.Name, p.Company, p.Price, p.OS);
}
```

Несмотря на то, что объект Smartphone никак не связан с таблицей Phones, при извлечении данных он также будет находится в наборе db.Phones, потому что наследование все равно будет действовать.

Назад **Содержание** **Вперед**