# LINQ to Entities

## Введение в LINQ to Entities

**Последнее обновление: 31.10.2015**

Ранее мы использовали ряд операций для получения данных из БД. В основе подобных операций лежит технология LINQ (Language Integrated Query), или точнее **LINQ to Entities**. LINQ to Entities предлагает простой и интуитивно понятный подход для получения данных с помощью выражений, которые по форме близки выражениям языка SQL.

Хотя при работе с базой данных мы оперируем запросами LINQ, но база данных понимает только запросы на языке SQL. Поэтому между LINQ to Entities и базой данных есть проводник, который позволяет им взаимодействовать. Этим проводником является провайдер **EntityClient**. Он создает интерфейс для взаимодействия с провайдером ADO.NET для SQL Servera.

Для начала взаимодействия с базой данных создается объект **EntityConnection**. Через объект **EntityCommand** он отправляет запросы, а с помощью объекта **EntityDataReader** считывает извлеченные из БД данные. Однако разработчику не надо напрямую взаимодействовать с этими объектами, фреймворк все сделает за него. Задача же разработчика сводится в основном к написанию запросов к базе данных с помощью LINQ.

Прежде чем приступить к обзору основных запросов в LINQ to Entities, для работы с материалом этой главы создадим новые модели по связи один-ко-многим:

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

    public ICollection<Phone> Phones { get; set; }
    public Company()
    {
        Phones = new List<Phone>();
    }
}

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

    public int CompanyId { get; set; }
    public Company Company { get; set; }
}
```

У нас здесь модель телефона и модель компании-производителя. Теперь создадим контекст данных и инициализатор базы данных начальными данными:

```
class PhoneContext : DbContext
{
    static PhoneContext()
    {
        Database.SetInitializer(new MyContextInitializer());
    }
    public PhoneContext() :base("DefaultConnection")
    {}

    public DbSet<Company> Companies { get; set; }
    public DbSet<Phone> Phones { get; set; }
}

class MyContextInitializer : DropCreateDatabaseAlways<PhoneContext>
{
    protected override void Seed(PhoneContext db)
    {
        Company c1 = new Company { Name = "Samsung" };
        Company c2 = new Company { Name = "Apple" };
        db.Companies.Add(c1);
        db.Companies.Add(c2);

        db.SaveChanges();

        Phone p1 = new Phone {Name="Samsung Galaxy S5", Price=20000, Company = c1};
        Phone p2 = new Phone {Name="Samsung Galaxy S4", Price=15000, Company = c1};
        Phone p3 = new Phone {Name="iPhone5", Price=28000, Company = c2};
        Phone p4 = new Phone {Name="iPhone 4S", Price=23000, Company = c2};

        db.Phones.AddRange(new List<Phone>(){p1, p2, p3, p4});
        db.SaveChanges();
    }
}
```

Чтобы база данных уже содержала некоторые данные, в инициализаторе бд создается несколько объектов. Чтобы задействовать инициализатор, он вызывается в статическом конструкторе контекста данных: `Database.SetInitializer(new MyContextInitializer());`

Для создания запросов в Linq to Entities, так же, как и в Linq to Objects, мы можем применять операторы LINQ и методы расширения LINQ.

Например, используем некоторые операторы LINQ:

```
using(PhoneContext db = new PhoneContext())
{
    var phones = from p in db.Phones
                 where p.CompanyId == 1
                 select p;
}
```

И тот же запрос с помощью методов расширений LINQ:

```
using(PhoneContext db = new PhoneContext())
{
        var phones = db.Phones.Where(p=> p.CompanyId == 1);
}
```

Оба запроса в итоге транслируются в одной выражение sql:

```
SELECT [Extent1].[Id] AS [Id],
           [Extent1].[Name] AS [Name],
           [Extent1].[Price] AS [Price],
           [Extent1].[CompanyId] AS [CompanyId]
FROM [dbo].[Phones] AS [Extent1]
WHERE 1 = [Extent1].[CompanyId]}
```

Важно понимать различие между Linq to Entities и Linq to Objects:

```
var phones = db.Phones.Where(p=> p.CompanyId == 1).ToList().Where(p=> p.Id<10);
```

Здесь используются два метода Where, но их реализация будет различной. В первом случае, `db.Phones.Where(p=> p.CompanyId == 1)` транслируется в выражение SQL, которое было рассмотрено выше. Далее метод `ToList()` по результатам запроса создает список в памяти компьютера. После этого мы уже имеем дело со списком в памяти, а не с базой данных. И далее вызов `Where(p=> p.Id<10)` будет обращаться к списку в памяти и будет представлять Linq to Object.

А теперь рассмотрим некоторые приемы применения LINQ к запросам из базы данных.

**Назад** **Содержание** **Вперед**