ADO.NET

(Entity Framework Core: Queries and Updates)

Артём Трофимушкин

EF Core

Платформа Entity Framework представляет собой набор технологий ADO.NET, обеспечивающих разработку приложений, связанных с обработкой данных.

Entity Framework (EF) Core — это кроссплатформенная и расширяемая ORM с открытым исходным кодом.

ORM (Object-Relational Mapping) — объектно-реляционное отображение, или преобразование) — технология, позволяющая связывать базы данных с концепциями объектно-ориентированных языков программирования.

ORM помогает работать с данными, как с объектами, т.е. на более высоком уровне, нежели подключения и SQL-запросы.

LINQ: Language Integrated Query

Особенность работы с Entity Framework заключается в использовании запросов LINQ для выборки данных из БД.

С помощью LINQ строятся похожие на SQL-запросы обращения к БД для извлечения данных в виде объектов.

Entities (сущности)

Основой всему в Entity Framework является понятие сущности (entity).

Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Сущности обладают свойствами. Свойства, однозначно определяющие конкретную сущность называются ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-к-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

DB Context (контекст базы данных)

Сущности входят в состав более крупной абстракции - контекста БД (собственно, самой БД).
В коде это выглядит так Пример:

```
public class SomeDbContext : DbContext
{
	private readonly string _connectionString;
	public DbSet<SomeEntity> Products { get; set; }
	public DbSet<AnotherEntity> Customers { get; set; }

	public OnlineStoreContext()
	{
		_connectionString = "...";
	}

	protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
	{
		optionsBuilder.UseSqlServer(_connectionString);
	}
}
```

Способы взаимодействия с БД

Code first — наиболее популярный подход для разработчиков, у которых хранилище — не в фокусе внимания на данной стадии разработки

- Высокая скорость, с которой разработчик создает хранилище. При этом нет необходимости думать о том, как там всё устроено на нижнем уровне? БД это просто хранилище, за работу которой отвечает EF.
- Изменения в БД управляются из кода, т.е. нет необходимости отдельно следить за версией БД (это не совсем истина, но разработчик может так себе это представлять).
- Ручные изменения в схеме БД недопустимы.

Способы взаимодействия с БД

Database first — очень популярный выбор, если БД разрабатывается параллельно профессиональными DBA, или если уже имеется существующая БД, с которой необходимо работать.

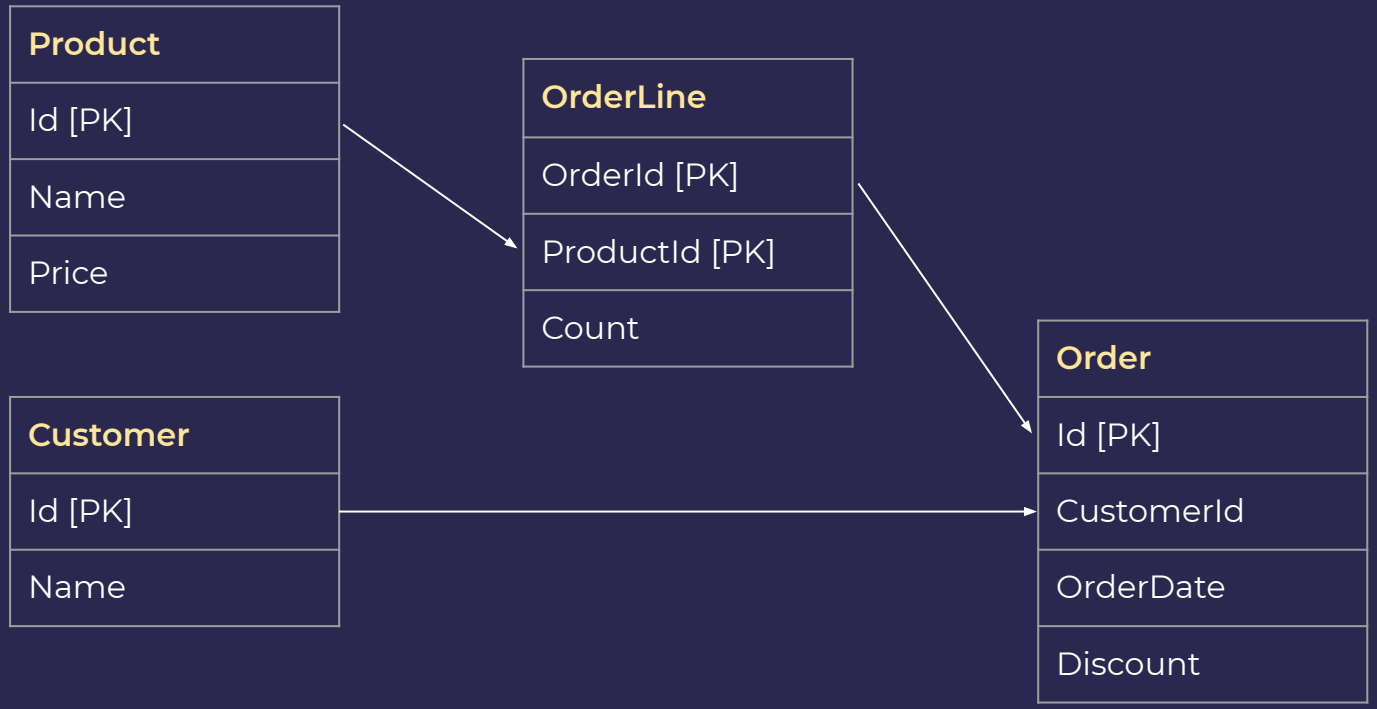
- Классы доступа к данным будут сгенерированы исходя из схемы БД.
- Вы можете продолжать вносить изменения в БД, обновляя ваши классы и внося в код необходимые изменения.

Совместная работа

Мы напишем приложение на базе EF Core на тему недавнего примера небольшого интернет-магазина.

Список таблиц:

- `Customer:` `Id, Name`
- `Product:` `Id, Name, Price`
- `Order:` `Id, CustomerId, OrderDate, Discount`
- `OrderLine:` `OrderId, ProductId, NumberOfItems`


Product
Id [PK]
Name
Price
OrderLine
OrderId [PK]
ProductId [PK]
Count
Customer
Id [PK]
Name
Order
Id [PK]
CustomerId
OrderDate
Discount

Вставка простых объектов

```
var customer = new Customer { Name = "Maria" };
using (var context = new OnlineStoreContext())
{
    context.Add(customer);
}
```

Теперь контекст *отслеживает* объект customer.

Однако самой вставки пока не происходит.

Вставка простых объектов

```
var customer = new Customer { Name = "Maria" };
using (var context = new OnlineStoreContext())
{
    context.Add(customer);
    context.SaveChanges();
}
```

Чтобы произошла вставка записи в БД, необходимо вызвать метод SaveChanges.

Что происходит внутри SaveChanges?

При выполнении метода SaveChanges контекст выполняет следующие шаги:

1. Проверяются все отслеживаемые контекстом объекты сущностей. Поскольку мы добавляли клиента (с помощью метода Add), контекст хранит информацию о том, что необходимо вставить новую запись в таблицу,
2. Подготавливается необходимый SQL-скрипт,
3. SQL-скрипт отправляется на выполнение в базу данных, причём обёрнутый в транзакцию.

Самостоятельная работа

Создаём несколько новых сущностей Product используя методы Add и AddRange соответствующего DbSet-а.

Упражняемся в специально отведенном для этого месте — методе InsertProducts

Самостоятельная работа (решение)

```
private static void InsertProducts()
{
     var product = new Product { Name = "Fenix 5 Plus Sapphire", Price = 73989.99M };
     using (var context = new OnlineStoreContext())
     {
          context.Products.Add(product);
          context.SaveChanges();
     }

     var products = new[]
     {
          new Product { Name = "Forerunner 645 Music", Price = 42199.99M },
          new Product { Name = "MARQ Aviator", Price = 208400 }
     };
     using (var context = new OnlineStoreContext())
     {
          context.AddRange(products);
          context.SaveChanges();
     }
}
```

Выборка данных с помощью EF Core

Особенность работы с Entity Framework заключается в использовании запросов LINQ (Language Integrated Query) для выборки данных из БД:

```
using (var context = new OnlineStoreContext())
{
    // select all the entities of type 'Customer'
    var customers = context.Customers.ToList();
}
```

С помощью LINQ строятся похожие на SQL-запросы обращения к БД для извлечения данных в виде объектов.

Два способа писать LINQ-запросы

LINQ-методы:

```
var allCustomers = context
      .Customers
      .ToList();
```

```
var severalCustomers = context
      .Customers
      .Where(c => c.Name == "Andrei")
      .ToList();
```

Синтаксис LINQ-запросов:

```
var allCustomersLinqSyntax = (
      from c
      in context.Customers
      select c
).ToList();
```

```
var severalCustomersLinqSyntax = (
      from c
      in context.Customers
      where c.Name == "Andrei"
      select c
).ToList();
```

Фильтрация объектов при выборке из БД

Если искомое значение вставляется константой прямо в лямбда-выражение:

```
...Where(c => c.Name == "Maria")
```

… в SQL-команду параметр НЕ добавляется:

```
SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE [c].[Name] = 'Maria'
```

Если искомее значение передаются в виде параметра:

```
var name = "Maria";

...Where(c => c.Name == name)
```

… в SQL-команду добавляется параметр:

```
@parameter = 'Maria'

SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE [c].[Name] = @parameter
```

LINQ to Entities Execution Methods

```
ToList()
First()   FirstOrDefault()
Single()  SingleOrDefault()
Last()*   LastOrDefault()*
Count()   LongCount()
Min()  Max()  Average()
```

Не LINQ-метод, однако DbSet его выполнит: `Find(keyValue)`

* Методы `Last` требуют, чтобы в запросе был `OrderBy`-метод, иначе из БД сначала вычитаются все данные, а потом вернётся последний элемент

Обновление объектов в БД

Для отслеживаемых контекстом объектов обновление произойдет автоматически:

```
using (var context = new OnlineStoreContext())
{
     var customer = context.Customers.First();
     customer.Name = "Mr. " + customer.Name;
     context.SaveChanges();
}
```

Для обновления неотслеживаемых объектов необходимо сначала вызвать метод Update:

```
using (var context = new OnlineStoreContext())
{
     var product = context.Products.AsNoTracking().First();
     product.Price *= 0.1M;
     context.Products.Update(product);
     context.SaveChanges();
}
```

Удаление объектов из БД

Для удаления необходимо иметь объект целиком, одного идентификатора недостаточно (несмотря на то, что в самом SQL-запросе ничего кроме идентификатора не фигурирует)

```
using (var context = new OnlineStoreContext())
{
     var customer = context.Customers.First();
     context.Customers.Remove(customer);
     context.SaveChanges();
}
```

Если очень хочется сделать удаление оптимальнее, можно составить необходимый SQL запрос:

```
context.Database.ExecuteSqlCommand(
     "DELETE FROM [dbo].[Customers] WHERE Id = {0}", 1);
```

Аналогично, можно выполнить любой SQL запрос

Полезные ссылки

- https://habr.com/ru/post/237889/
- https://ru.stackoverflow.com/questions/718991/Практическая-разница-между-подходами-к-наследованию-в-entity-framework-при-разра
- https://docs.microsoft.com/ru-ru/ef/core/modeling
- https://docs.microsoft.com/en-gb/ef/core/miscellaneous/cli/dbcontext-creation
- https://www.entityframeworktutorial.net/code-first/column-dataannotations-attribute-in-code-first.aspx

Домашняя работа

На основе материалов урока 27 (база данных корреспонденции) реализовать модель (в терминах EF) и контекст к этой базе данных

Спасибо за внимание.