



ADO.NET

(Entity Framework Core: Migrations)

Андрей Голяков

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 ЕФ.
- **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.



EF Core Migrations

Модель данных в процессе разработки может измениться и перестанет соответствовать базе данных.

Вы всегда можете удалить базу данных, и EF создаст для вас новую версию, в точности соответствующую модели, однако такая процедура приводит к потере текущих данных.

Функция **миграции** в EF Core позволяет **последовательно применять изменения схемы** к базе данных, чтобы синхронизировать ее с моделью данных в приложении **без потери существующих данных**.



Создание и применение Миграций

С миграциями работаем в окне **Package Management Console**:
View > Other Windows > Package Management Console.

- Команда **Add-Migration** создаёт миграцию — код, который сгенерирует SQL-скрипт на изменение схемы данных БД согласно модели данных (нашему контексту БД).
- Команда **Script-Migration** генерирует и показывает SQL-скрипт.
- Команда **Update-Database** применяет указанную миграцию (запускает подготовленный SQL-скрипт).
- Полный список команд можно посмотреть с помощью команды **Get-Help EntityFrameworkCore**



Рекомендации для миграций

Миграции можно применять двумя способами - через PS-команду и просто запуском сгенерированного SQL-запроса.

Для среды разработки рекомендуется накатывать миграции используя PS-команду `update-database`.

Для production-среды рекомендуется использовать **SQL-скрипт**.

Обратите внимание, что в первый раз базы данных не существует. Когда миграция накатывается через PS-команду `update-database`, БД будет создана автоматически.

Если же мы накатываем SQL-скрипт вручную, необходимо предварительно создать пустую базу данных!



Совместная работа

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

Вспоминаем схему БД `OnlineStore`. У нас было 4 таблицы:

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

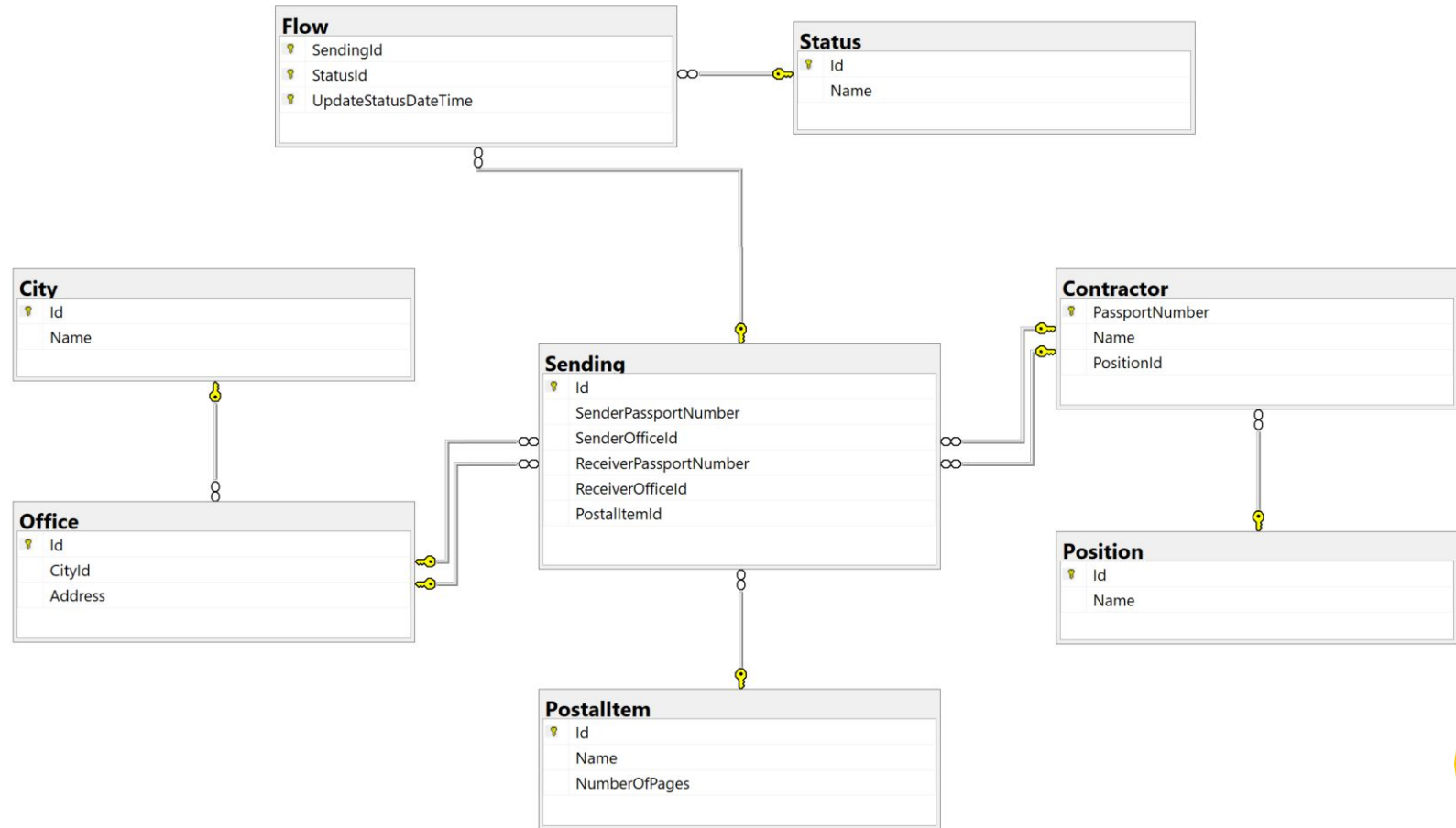


Самостоятельная работа

1. Спроектировать классы для сущностей нашей БД корреспонденции (на которой мы тренировались в приведении БД к 3-ей нормальной форме)
2. Создать первую миграцию с именем `InitialCreate`
3. Инициализировать схему БД `CorrespondenceEF`



Correspondence database schema



Совместная работа

Демонстрация управления создаваемой схемой с помощью

- Data Annotation атрибутов
- Fleunt API



Самостоятельная работа

С помощью Data Annotation атрибутов и/или Fluent API добиться максимального соответствия схемы CorrespondenceEF, создаваемой EF Core, оригинальной схеме БД Correspondence, разработанной на уроке (скрипты прилагаются).



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

С помощью Data Annotation атрибутов и/или Fluent API добиться максимального соответствия схемы CorrespondenceEF, создаваемой EF Core, оригинальной схеме БД Correspondence, разработанной на уроке (скрипты прилагаются).



Спасибо за внимание.

