Введение

DSD.NAP.9.6.37-Intro

# Введение

Этот материал является частью документации к стартовому комплексу NET-разработчика «Точка сборки» (NET Assembly Point, NAP)

Стартовый комплекс «Точка сборки» в основном предназначен для упрощения разработки приложений, использующих Entity Framework CodeFirst для маппинга базы данных, при этом однако содержит NET-пакеты более общего назначения, например ..\Hosting и ..\Logger.

Структура комплекса опирается на концепцию Onion-архитектуры, что, тем не менее, не накладывает никаких требований на архитектуру разрабатываемого приложения. Напротив, именно технологические решения приложения определяют конфигурацию комплекса.

# Entity Framework

Entity Framework (EF) — это объектно-реляционный сопоставитель (ORM, Object-Relational Mapper) для .NET, который упрощает работу с базами данных, позволяя разработчикам взаимодействовать с данными через объекты .NET. EF является мощным фреймворком, предлагающим различные технологии конфигурирования маппинга, которые, в общем случае, делятся на две основные концепции: DatabaseFirst и CodeFirst.

## Основные концепции Entity Framework

Первая концепция, DatabaseFirst, предполагает предварительное создание базы данных, включая все необходимые связи, индексы и атрибуты. Далее ORM генерирует классы .NET, соответствующие таблицам базы данных. Эта концепция особенно удобна для платформ баз данных с графическими редакторами ER-диаграмм, которые позволяют легко построить полную структуру базы данных. В отсутствие такого редактора создание базы данных сводится к реализации языковых методов платформы, что, по сути, приводит к аналогу CodeFirst, реализуемой другими средствами. Кроме того, DatabaseFirst может оказаться крайне неудобной в случае рефакторинга базы данных.

Более современная концепция CodeFirst позволяет разработчикам определять модели данных (Domain Model) с помощью кода C# или VB.NET до создания базы данных. На основе Domain Model фреймворк создает таблицы базы данных и связи между ними.

Значительным преимуществом CodeFirst является простота рефакторинга в единой языковой среде .NET, включая возможности миграции базы данных без нарушения ее целостности.

## Entity Framework CodeFirst

Code First предлагает несколько способов определения моделей данных и связей между сущностями:

* **Convention over Configuration (Соглашение по конфигурации):** EF автоматически определяет связи и свойства на основе соглашений об именовании. Это самый простой подход.
* **Fluent API:** более гибкий и мощный способ настройки маппинга с по- мощью специального API.
* **Атрибуты данных:** Атрибуты, применяемые к классам и свойствам модели, позволяют задавать конфигурацию маппинга.
* **Навигационные свойства:** Свойства в классах модели, представляющие связи между сущностями.

Методы можно комбинировать, начиная с простых решений и переходя к более сложным по мере необходимости.

## Контекст базы данных

В Code First модели данных реализуются через контекст базы данных, для чего создается класс, наследующий от абстрактного класса DbContext, который входит в состав базового пакета Microsoft.EntityFrameworkCore.

Контекст настраивается путем подключения провайдера для конкретной платформы базы данных (например, Microsoft.EntityFrameworkCore.SqlServer) и указания параметров подключения. Затем в контекст добавляются свойства типа DbSet<T>, представляющие коллекцию сущностей (entity), где Т является собой тип сущности.

После этого можно создавать экземпляры контекста и использовать их для взаимодействия с базой данных, причем создание первого экземпляра контекста одновременно создает и собственно базу данных.

Ниже приведен пример класса контекста:

using Microsoft.EntityFrameworkCore;

namespace Examples;

public class ApplicationDbContext: DbContext

{

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) => optionsBuilder.UseSqlServer(@"Server=(localdb)\MSSQLLocalDB; Initial Catalog=dbCobra.9.6.36.6; Integrated Security=True");

public DbSet<Foo> Foos { get; set; }

public DbSet<Bar> Bars { get; set; }

}

# Назначение, цель и задачи EFCFS

Цель разработки стартового комплекса EFCFS – очистка кода приложения от рутинных процедур маппинга, хостинга и логинга за счет их переноса в абстрактные объекты или реализованные экземпляры классов, размещенных в пакетах комплекса.

В целом, структура стартового комплекса пакетов EFCFS согласована с Onion-архитектурой - каждому слою (layer) соответствует один или более базовых пакетов, содержащих абстрактные объекты, реализация которых обеспечивает генерацию схемы и объектов базы данных, а также регистрацию ряда стартовых сервисов. Базовые пакеты объединены в более общие интегрированные пакеты.

public class Foo1

{

public int Id { get; set; } // Первичный ключ

}

public class Bar1

{

public int Id { get; set; } // Первичный ключ

public int Foo1Id { get; set; } // Внешний ключ

}