# ПОДХОДЫ CODE FIRST, DATABASE FIRST И MODEL FIRST В СОЗДАНИИ БАЗЫ ДАННЫХ ИСПОЛЬЗУЯ ENTITY FRAMEWORK

**Роман КИОРОГЛО**

*Департамент Программной Инженерии и Автоматики, группа TI-192 F/R, Факультет Вычислительной Техники, Информатики и Микроэлектроники, Технический Университет Молдовы, Кишинев, Республика Молдова*

Автор корреспондент: Роман КИОРОГЛО, e-mail: chioroglo.roman@isa.utm.md

**Научный руководитель: Дориан САРАНЧУК,** DISA, FCIM, UTM

***Резюме.*** *При разработке на платформе .NET, во время этапа создания базы данных, происходит выбор между несколькими подходами к её созданию, а именно, Code First, Database First и Model First. Каждый из подходов имеет свои приемущества и недостатки перед другими. В данной работе описываются подходы, их отличия и случаи, в которых можно применить каждый из них.*

***Ключевые слова****: Базы данных, .NET, Проектирование базы данных, Модель состояния, классы .NET, SQL, миграции*

## Введение

При создании базы данных в платформе .NET у разработчика возникает вопрос, какой использовать подход? В Entity Framework существуют три основных подхода для создания базы данных:

Подход Code First предполагает создание базы данных на основе модели данных, которая описывается в коде приложения. На основе этой модели автоматически генерируется схема базы данных. Этот подход подходит для проектов, в которых необходимо быстро начать работу с проектом, не тратя много времени на описание модели данных [1].

Подход Database First предполагает создание базы данных на основе существующей схемы базы данных. Для этого необходимо создать файл модели EDMX на основе существующей базы данных. Этот подход подходит для проектов, в которых есть существующая база данных, с которой нужно работать.

Подход Model First предполагает создание базы данных на основе визуальной модели, которая описывается в Entity Framework Designer. На основе этой модели автоматически генерируется схема базы данных. Этот подход подходит для проектов, в которых необходимо быстро создать модель данных без необходимости создания базы данных с нуля [1].

Каждый подход имеет свои преимущества и недостатки, и выбор зависит от конкретных требований проекта и предпочтений команды разработчиков.

## Code First

Code First - это один из подходов к разработке базы данных в Entity Framework, в котором разработка базы данных начинается с создания классов в языке программирования и затем преобразования этих классов в таблицы базы данных. Этот подход позволяет разработчикам сосредоточиться на разработке объектной модели, а не на проектировании базы данных.

Основные шаги в подходе Code-First:

1. Определение классов: Разработчики определяют классы для каждой таблицы в базе данных. Каждый класс представляет сущность в базе данных и имеет свойства, соответствующие столбцам в таблице.

2. Определение отношений: Разработчики определяют отношения между классами, используя аннотации или Fluent API. Отношения могут быть один-к-одному, один-ко-многим или многие-ко-многим.
3. Контекст данных: Разработчики определяют контекст данных, который настроен для работы с базой данных. Контекст данных включает в себя определения классов, отношений и настроек соединения с базой данных.
4. Создание базы данных: После определения классов и контекста данных, Entity Framework создает базу данных на основе этих определений. Разработчики могут выбрать, где хранить базу данных (локально или на сервере).
5. Изменение базы данных: После создания базы данных, разработчики могут использовать механизм миграции для внесения изменений в базу данных, когда требуется изменение модели.

Code-First предоставляет гибкость в разработке базы данных и позволяет разработчикам изменять модель базы данных без необходимости изменения схемы базы данных. Этот подход также обеспечивает лучшую поддержку для разработки с использованием практик Agile, таких как быстрые итерации и TDD (разработка через тестирование) [2].

Рассмотрим пример использования подхода Code First для создания модели данных для простого приложения, которое хранит информацию о книгах в библиотеке.

Определение классов модели данных:

csharp
Copy code

```
public class Book
{
   public int Id { get; set; }
   public string Title { get; set; }
   public string Author { get; set; }
   public int Year { get; set; }
}

public class LibraryContext : DbContext
{
   public DbSet<Book> Books { get; set; }
}
```

Класс Book определяет модель для книги, которая содержит свойства Id, Title, Author и Year. Класс LibraryContext наследуется от класса DbContext и содержит свойство Books, которое представляет коллекцию объектов Book в базе данных [2].

Конфигурация модели данных:

csharp
Copy code

```
public class LibraryContext : DbContext
{
   public LibraryContext() : base("name=LibraryContext")
   {
   }
   protected override void OnModelCreating(DbModelBuilder modelBuilder)
   {
      modelBuilder.Entity<Book>()
         .Property(b => b.Title)
         .IsRequired()
         .HasMaxLength(100);
```

```
        modelBuilder.Entity<Book>()
            .Property(b => b.Author)
            .IsRequired()
            .HasMaxLength(50);
    }
    public DbSet<Book> Books { get; set; }
}
```

В методе OnModelCreating мы можем использовать Fluent API для определения свойств и настроек для нашей модели данных. В данном случае мы устанавливаем обязательное наличие свойств Title и Author и ограничения на максимальную длину.

Создание базы данных:

```
csharp
Copy code
using (var context = new LibraryContext())
{
    context.Database.CreateIfNotExists();
}
```

Создание базы данных происходит автоматически при первом обращении к контексту данных.

Таким образом, подход Code First позволяет определять модель данных с помощью классов и свойств, а затем создавать базу данных на основе этой модели. Благодаря этому подходу разработчик может иметь большую гибкость при работе с моделью данных и управлении ее изменениями.

**Database First**

Database First - это один из подходов к разработке базы данных в Entity Framework, в котором база данных уже существует, и классы создаются автоматически на основе существующей схемы базы данных. Этот подход позволяет быстро создавать объектную модель на основе существующей базы данных, что может быть полезно, если база данных уже существует или если проектирование базы данных является первоочередным приоритетом.

Основные шаги в подходе Database First:

1. Создание модели из базы данных: Разработчики используют визуальный дизайнер модели Entity Framework или командную строку, чтобы создать модель базы данных на основе существующей базы данных.
2. Генерация классов: После создания модели базы данных Entity Framework автоматически генерирует классы, которые соответствуют таблицам в базе данных. Эти классы могут быть использованы для доступа к данным в базе данных.
3. Контекст данных: Разработчики создают контекст данных, который настроен для работы с базой данных. Контекст данных включает в себя определения классов, отношений и настроек соединения с базой данных.
4. Использование объектной модели: Разработчики могут использовать объектную модель для доступа к данным в базе данных и для выполнения запросов, обновления и вставки данных.

Database First позволяет быстро создавать объектную модель на основе существующей базы данных и предоставляет возможность использовать уже существующую базу данных без необходимости создания модели с нуля. Однако этот подход может быть менее гибким, чем Code-First, когда требуется внести изменения в схему базы данных.

Рассмотрим пример использования подхода Database First для создания модели данных для того же приложения, которое хранит информацию о книгах в библиотеке.

1. Создание базы данных
2. Сначала мы создадим базу данных, содержащую таблицу Books, которая будет использоваться в нашем приложении. Это можно сделать с помощью SQL скрипта или визуального редактора баз данных, такого как Microsoft SQL Server Management Studio.
3. Генерация модели данных
4. Затем мы можем использовать встроенный инструмент в Entity Framework для генерации модели данных на основе существующей базы данных. Для этого нужно выполнить следующие шаги:
   - Добавить новый элемент модели в проект, используя контекстное меню проекта в Visual Studio.
   - В диалоговом окне добавления элемента выбрать "ADO.NET Entity Data Model" и дать ему имя.
   - Выбрать "EF Designer from database" в качестве типа модели и нажать кнопку "Next".
   - Указать строку подключения к базе данных и выбрать таблицы, которые мы хотим использовать в нашей модели данных. В нашем случае это таблица Books.
   - Нажать кнопку "Finish", чтобы сгенерировать модель данных.
5. Использование модели данных

После генерации модели данных мы можем использовать ее в нашем приложении. Например, мы можем получить список всех книг из базы данных следующим образом:

```
csharp
Copy code
using (var context = new LibraryEntities())
{
   var books = context.Books.ToList();
   foreach (var book in books)
   {
      Console.WriteLine($"{book.Id} - {book.Title} by {book.Author}, published in {book.Year}");
   }
}
```

Здесь LibraryEntities - это имя контекста данных, который был сгенерирован вместе с моделью данных. Мы можем использовать его для доступа к данным в базе данных.

**Model First**

Model First - это один из подходов к разработке базы данных в Entity Framework, в котором разработчики создают модель данных с помощью визуального дизайнера и затем используют эту модель для генерации базы данных и классов Entity Framework. Этот подход удобен, когда требуется создать базу данных, но отсутствует готовая схема или требуется создать модель данных на основе предметной области.

Основные шаги в подходе Model First:

1. Создание модели: Разработчики используют визуальный дизайнер модели Entity Framework для создания модели данных, которая представляет сущности, отношения и атрибуты в предметной области.
2. Генерация базы данных и классов: После создания модели Entity Framework автоматически генерирует базу данных и классы, которые соответствуют модели. Эти классы могут быть использованы для доступа к данным в базе данных.
3. Контекст данных: Разработчики создают контекст данных, который настроен для работы с базой данных. Контекст данных включает в себя определения классов, отношений и настроек соединения с базой данных.

4. Использование объектной модели: Разработчики могут использовать объектную модель для доступа к данным в базе данных и для выполнения запросов, обновления и вставки данных.

Model First позволяет разработчикам быстро создавать модель данных на основе предметной области и генерировать базу данных и классы на основе этой модели. Этот подход удобен для разработки приложений с небольшой или средней сложностью, где требуется быстрый старт без особого уделяя внимания разработке схемы базы данных. Однако он может быть менее гибким, чем Code-First, когда требуется внести изменения в схему базы данных.

**Выводы**

Выбор подхода к работе с Entity Framework (Code First, Database First или Model First) зависит от конкретной задачи и требований к проекту. Рассмотрим некоторые рекомендации по выбору подхода.

Code First

Подход Code First подходит для проектов, в которых:

- Необходимо создать базу данных с нуля.
- Нужна гибкость в определении модели данных.
- Есть предпочтение использованию кода для описания модели данных вместо визуального редактора.
- Проект разрабатывается командой разработчиков, которые знакомы с принципами объектно-ориентированного программирования.

Database First

Подход Database First подходит для проектов, в которых:

- Есть существующая база данных, с которой нужно работать.
- Необходимо быстро начать работу с проектом, не тратя много времени на описание модели данных.
- Необходимо работать с базой данных, которая разработана вне проекта.
- Необходимо работать с базой данных, которая поддерживается другими системами.

Model First

- Подход Model First подходит для проектов, в которых:
- Нужно быстро создать модель данных без необходимости создания базы данных с нуля.
- Нужно создать визуальное представление модели данных, которое будет использоваться в проекте.
- Есть возможность создать модель данных визуально, используя Entity Framework Designer.
- Есть необходимость быстро изменять структуру базы данных, не затрагивая код.

В целом, выбор подхода зависит от конкретных требований проекта и предпочтений команды разработчиков. Каждый подход имеет свои преимущества и недостатки, и выбор зависит от конкретной ситуации.

**Библиография**

1. Tutorial: Get Started with Entity Framework 6 Code First using MVC 5. [online] [дата обращения 02.03.2023], Доступно: https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application
2. Database Initialization in Entity Framework 6. [online] [дата обращения 02.03.2023], Доступно: https://www.entityframeworktutorial.net/code-first/database-initialization-in-code-first.aspx