Турсунов Баходурхон

**1. Объясните концепцию миграций в Entity Framework Core. Как миграции помогают**

**синхронизации модели EF Core и схемы базы данных?**

Миграции в EF Core — это способ управления изменениями в модели данных и синхронизации их со схемой базы данных. Они позволяют:

Отслеживать изменения: EF Core отслеживает изменения в моделях сущностей и создает код миграции, отражающий эти изменения.

Применять изменения: Код миграции может быть применен к базе данных для обновления ее схемы.

Откатывать изменения: можно откатить миграции, чтобы вернуться к предыдущей версии схемы.

Синхронизация модели и схемы:

EF Core сравнивает текущую модель данных с информацией о схеме, хранящейся в специальной таблице миграций.

На основе этого сравнения создается код миграции, содержащий инструкции для обновления схемы (например, добавление таблиц, столбцов, индексов).

Применение миграции к базе данных приводит к синхронизации схемы с моделью.

**2. Создайте простую модель для блог-приложения, включающую как минимум две сущности: Blog и Post. Реализуйте миграцию для инициализации схемы базы данных для этих сущностей.**

1. public class Blog

2. {

3. public int BlogId { get; set; }

4. public string Url { get; set; }

5.

6. public List<Post> Posts { get; set; }

7. }

8.

9. public class Post

10. {

11. public int PostId { get; set; }

12. public string Title { get; set; }

13. public string Content { get; set; }

14.

15. public int BlogId { get; set; }

16. public Blog Blog { get; set; }

17. }

Создание миграции:

Установить пакет Microsoft.EntityFrameworkCore.Tools

Выполнить команду: dotnet ef migrations add InitialCreate

EF Core создаст файл миграции с инструкциями для создания таблиц Blogs и Posts.

**3. Опишите роль LINQ в запросе данных с помощью Entity Framework Core. Как EF Core**

**переводит LINQ-запросы в язык запросов, специфичный для базы данных?**

LINQ (Language Integrated Query) — это мощный инструмент для запроса данных в C#. EF Core использует LINQ для создания запросов к базе данных:

LINQ-выражения: Вы пишете запросы на C# с использованием знакомого синтаксиса.

Перевод в SQL: EF Core переводит LINQ-выражения в SQL-запросы, специфичные для используемой базы данных.

Выполнение запросов: SQL-запросы выполняются на сервере базы данных, а результаты возвращаются в приложение.

**4. Используя модель, созданную в вопросе 2, напишите LINQ-запрос для получения всех постов, связанных с определенным блогом. Продемонстрируйте, как загрузить связанные данные с помощью метода .Include().**

1. // Получить блог с заданным ID

2. var blog = context.Blogs

3. .Include(b => b.Posts)

4. .FirstOrDefault(b => b.BlogId == 1);

5.

6. // Доступ к постам блога

7. var posts = blog.Posts;

8.

**5. Обсудите два основных подхода к сохранению данных в базе данных с помощью EF Core. Что такое "единица работы" и как она реализована в EF Core?**

Подходы к сохранению:

Connected: Изменения отслеживаются DbContext, а затем сохраняются в базу данных вызовом SaveChanges().

Disconnected: Изменения вносятся в отсоединенные сущности, а затем эти сущности прикрепляются к DbContext и сохраняются.

Единица работы:

Представляет собой набор операций с данными, которые должны быть выполнены как единое целое.

В EF Core единица работы реализована DbContext.

SaveChanges() фиксирует все изменения, отслеживаемые DbContext, в базе данных.

**6. Добавьте в блог-приложение функцию, позволяющую обновлять свойство Url сущности Blog.**

1. // Получить блог

2. var blog = context.Blogs.Find(1);

3.

4. // Изменить URL

5. blog.Url = "newurl.com";

6.

7. // Сохранить изменения

8. context.SaveChanges();

9. сущности. Покажите, как отслеживаются и сохраняются изменения с помощью DbContext.

10.

**7. Каково назначение методов DbSet<TEntity>.Add и DbSet<TEntity>.Remove в**

**EF Core? Приведите сценарии использования каждого из них.**

DbSet<TEntity>.Add(entity): добавляет новую сущность в контекст и помечает ее для вставки в базу данных.

DbSet<TEntity>.Remove(entity): помечает сущность для удаления из базы данных.

Сценарии использования:

Add: Добавление новых записей в таблицу.

Remove: Удаление существующих записей.

**8. Реализуйте функцию добавления нового поста в блог и функцию удаления поста.**

**Используйте методы Add и Remove, продемонстрированные в вашем коде.**

1. // Добавить новый пост

2. var newPost = new Post { Title = "Новый пост", Content = "..." };

3. context.Posts.Add(newPost);

4. context.SaveChanges();

5.

6. // Удалить пост

7. var postToDelete = context.Posts.Find(1);

8. context.Posts.Remove(postToDelete);

9. context.SaveChanges();

10.

В этих примерах Add и Remove используются для управления записями в таблице Posts.