Manish Jaiswal

EF Core Series

MASTERING

EF Core Database First Approach

swipe

Manish Jaiswal

EF Core Series

01

Introduction

Database First Approach allows you to create a data model based on an existing database. This approach is ideal for projects where the database is already designed and running.

Key Features:

- Works with existing databases
- Auto-generates context and entity classes
- Supports complex database structures

swipe

Manish Jaiswal

EF Core Series

02

Reverse Engineering an Existing Database

To use the Database First Approach, you need to reverse engineer your existing database. Use the Scaffold-DbContext command to generate your context and entity classes.

```
// This command will create a Models directory with
// context and entity classes representing your database.
Scaffold-DbContext "YourConnectionString"
    Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

swipe

Manish Jaiswal

EF Core Series

03

Customizing the Generated Models

The generated models can be **customized** to fit your application's needs. You can modify data annotations, relationships, and other properties.

swipe

04

Example

```
1   public partial class Product
2   {
3       [Key]
4       public int ProductId { get; set; }
5
6       [Required]
7       [MaxLength(50)]
8       public string ProductName { get; set; }
9
10      public decimal Price { get; set; }
11  }
```

Here, we've added data annotations to the **Product** model for validation and constraints.

swipe

Manish Jaiswal

EF Core Series

05

Managing Changes with Migrations

Even with Database First, you can manage schema changes using migrations. This allows you to keep your database schema in sync with your model changes.

Example: Adding a Migration

```
1   Add-Migration AddNewColumn
2   Update-Database
```

swipe

Manish Jaiswal

EF Core Series

06

Conclusion and Best Practices

- Regularly update your models by re-scaffolding as the database evolves.
- Keep your models clean and organized by splitting them into partial classes.
- Use migrations to handle schema changes efficiently.

swipe