# INTRODUCTION TO ENTITY FRAMEWORK

Moein Nemati

# TABLE OF CONTENTS

# 01 WHAT IS ENTITY FRAMEWORK?

Introduction to Entity Framework
and ORM concepts.

**Entity Framework (EF)** is an **Object-Relational Mapping (ORM)** framework developed by Microsoft. It enables developers to work with data in a more **object-oriented** way, allowing you to use objects in your code that map to **rows** in a database table.

**ORM** abstracts and handles all database interactions for you, meaning you **rarely** have to write SQL code directly, making database management far more intuitive for developers familiar with **object-oriented** programming.

# ENTITY FRAMEWORK

# 02 FEATURES OF ENTITY FRAMEWORK

Entity Framework offers a range of features that cater to various needs and scenarios, enhancing its flexibility and functionality in application development.

# APPROACHES TO MODELING

Entity Framework supports three major approaches to model your data:

**Code-First Approach**

Start by coding C# classes, and then generate the database schema from these models using migrations. Ideal for projects without a pre-existing database.

**Database-First Approach**

Generate C# data models from an existing database schema, perfect for integrating with current systems.

**Model-First Approach**

Create models visually using a designer tool in Visual Studio, and then generate both the database schema and the code.

# QUERYING DATA

Entity Framework enhances data querying with robust support for LINQ, offering two syntaxes:

### 1. LINQ Query Syntax

Similar to SQL, it provides a readable and expressive way to write queries.

```
var blogs = from b in context.Blogs
            where b.Name.Contains("tech")
            select b;
```

### 2. LINQ Method Syntax (Extension Methods)

Uses extension methods and is more flexible for composing queries programmatically.

```
var blogs = context.Blogs.Where(b =>
b.Name.Contains("tech")).ToList();
```

# 03

# CRUD OPERATIONS SIMPLIFIED

Demonstrating CRUD operations using Entity Framework.

# CRUD OPERATIONS

CRUD operations—Create, Read, Update, Delete—are fundamental for any application interacting with a database, and EF makes these operations straightforward.

# EXAMPLES

| | | |
|---|---|---|
| **CREATE** | `var newBlog = new Blog { Name = "New Blog" };`<br>`context.Blogs.Add(newBlog);`<br>`context.SaveChanges();` | `INSERT INTO blogs (name) VALUES ('New Blog');` |
| **READ** | `var blogs = context.Blogs.ToList();` | `SELECT * FROM blogs;` |
| **UPDATE** | `var blog = context.Blogs.Find(1);`<br>`blog.Name = "Updated Blog Name";`<br>`context.SaveChanges();` | `UPDATE blogs SET name = 'Updated Blog Name' WHERE id = 1;` |
| **DELETE** | `var blog = context.Blogs.Find(1);`<br>`context.Blogs.Remove(blog);`<br>`context.SaveChanges();` | `DELETE FROM blogs WHERE id = 1;` |

# 04

# COMPARISON OF USING AN ORM VS. DIRECT SQL

The decision to use Entity Framework involves weighing both pros and cons.

# ADVANTAGES

1. **Simplification of Code:** Automates tasks such as generating SQL queries.
2. **Improved Productivity:** Faster development cycles by focusing on business logic rather than database intricacies.
3. **Data Consistency:** ORM ensures consistent application of data access rules and business logic.

# DISADVANTAGES

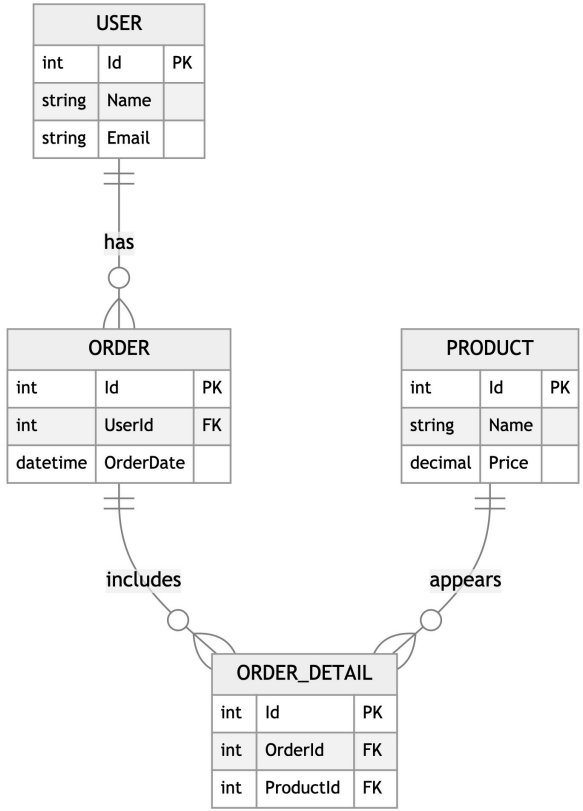
1. **Performance Overhead:** Sometimes ORM can be slower than optimized SQL, especially with complex queries.

2. **Less Control:** Developers have less control over the executed SQL, which can be problematic for fine-tuned performance optimization.

# 05 PRACTICAL EXAMPLE

# EXAMPLE OVERVIEW

This practical example demonstrates an application that manages products, orders, and users with various functionalities implemented both with and without Entity Framework Core. Here's how each functionality is handled using EF Core versus traditional SQL approaches:



# 06 CONCLUSION

Final thoughts and encouragement for further exploration.

# WRAPPING UP

In conclusion, **Entity Framework** provides a robust framework for managing database operations with minimal **direct SQL coding**, leveraging the power of .NET's **object-oriented** features.

While it simplifies many aspects of **database management** and increases **development speed**, it is also essential to consider the potential **performance implications** and **control limitations**.

I encourage you to experiment with **Entity Framework** to better understand its **capabilities** and determine when it is the most appropriate tool for your projects.

# RESOURCES

- Entity Framework documentation hub
- Entity Framework Tutorial