**ENTITY FRAMEWORK CORE**

**What is Entity Framework (EF)?**

* Entity Framework (EF) is a tool in .NET Core that helps developers interact with databases using **C# code** instead of writing SQL queries directly.
* It’s called an **ORM (Object-Relational Mapping)** tool.  
  **Why?**
  + It converts data in the database (tables, rows) into objects (classes, objects) in C#.

**Why use Entity Framework?**

1. **Reduces SQL Code**
   * You don’t need to write SQL queries for basic CRUD (Create, Read, Update, Delete) operations.  
     Example: Instead of SELECT \* FROM Products, EF lets you use:

var products = dbContext.Products.ToList();

1. **Easier to Maintain**
   * Your code becomes cleaner and easier to manage because it’s all written in C#.
2. **Works with Multiple Databases**
   * EF can connect to different databases like SQL Server, MySQL, PostgreSQL, etc.

**How Does EF Work?**

1. **Database Context**
   * The Database Context (DbContext) acts like a bridge between your application and the database.
   * It contains **DbSet** properties for each table in the database.  
     Example:

public class AppDbContext : DbContext

{

public DbSet<Product> Products { get; set; }

}

1. **Models**
   * Classes represent tables in the database. These classes are called **models**.  
     Example: A Products table in the database can be represented by a Product class:

public class Product{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

}

**Approaches in EF**

1. **Code-First**
   * Write the C# code (models) first, and EF will create the database for you.
2. **Database-First**
   * Use an existing database, and EF will generate the C# classes (models) for you.

### ****Common EF Operations****

EF is commonly used for **CRUD Operations** (Create, Read, Update, Delete).

#### **1. Create (Insert)**

Add a new record to the database.

var product = new Product { Name = "Laptop", Price = 50000 };

dbContext.Products.Add(product);

dbContext.SaveChanges();

#### **2. Read (Select)**

Get data from the database.

// Get all products

var products = dbContext.Products.ToList();

// Get a specific product by Id

var product = dbContext.Products.FirstOrDefault(p => p.Id == 1);

#### **3. Update**

Update an existing record in the database.

var product = dbContext.Products.FirstOrDefault(p => p.Id == 1);

if (product != null)

{

product.Price = 55000;

dbContext.SaveChanges();

}

#### **4. Delete**

Remove a record from the database.

var product = dbContext.Products.FirstOrDefault(p => p.Id == 1);

if (product != null)

{

dbContext.Products.Remove(product);

dbContext.SaveChanges();

}

### ****How to Set Up EF in .NET Core****

#### **Step 1: Install EF Core**

Run the following commands in the **Package Manager Console** or **Terminal**:

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

#### **Step 2: Add DbContext**

Create a class that inherits from DbContext:

public class AppDbContext : DbContext

{

public DbSet<Product> Products { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer("YourConnectionStringHere");

}

}

#### **Step 3: Use Migrations (For Code-First Approach)**

1. Add migration (create database structure):

Add-migration anyname

1. Apply migration (create database in SQL Server):

Update-database

**Use Scaffolding (For Database-First Approach)**

1. Scaffold-DbContext "YourConnectionStringHere" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models