EF Core 8.0 HOL

Lab 1: Understanding ORM with a Retail Inventory System

### What is ORM?

### ****Object-Relational Mapping (ORM)**** is a technique that allows developers to interact with a relational database using object-oriented programming languages like C#.

**In EF Core**, this means:

Classes map to tables

Properties map to columns

Relationships (e.g., one-to-many) are managed using navigation properties

EF Core vs EF Framework:

Entity Framework Core (EF Core) is a modern, cross-platform ORM designed to work with .NET Core and newer .NET versions like .NET 5/6 and beyond. It offers better performance, particularly through features like compiled queries and improved change tracking, making it well-suited for modern, lightweight applications and microservices. EF Core supports modern development practices such as asynchronous programming, LINQ, and compiled models. In contrast, Entity Framework 6 (EF6) is built exclusively for the Windows-based .NET Framework and is primarily used in legacy or older Windows desktop and web applications. While EF6 is mature and considered more stable for long-established projects, it lacks the performance and flexibility enhancements introduced in EF Core.

EF Core 8.0 Features:

**JSON Column Mapping**: Store and query structured JSON data inside a single column (e.g., SQL Server nvarchar(max)).

**Compiled Models**: Pre-compiling the EF model boosts startup performance.

**Interceptors**: Hook into EF Core operations for logging, validation, etc.

**Bulk Operations**: More efficient updates/deletes for large datasets.

Create a .NET Console App:

Product.cs

namespace RetailInventory.Models;

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public int StockLevel { get; set; }

public int CategoryId { get; set; }

public Category Category { get; set; }

}

Category.cs

namespace RetailInventory.Models;

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

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

AppDbContext.cs

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

namespace RetailInventory.Data;

public class AppDbContext : DbContext

{

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

public DbSet<Category> Categories { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=RetailInventoryDb;Trusted\_Connection=True;");

}

}

Program.cs

using RetailInventory.Data;

using RetailInventory.Models;

using Microsoft.EntityFrameworkCore;

// Ensure DB is created

using (var context = new AppDbContext())

{

context.Database.EnsureCreated();

if (!context.Categories.Any())

{

// Seed categories and products

var electronics = new Category { Name = "Electronics" };

var groceries = new Category { Name = "Groceries" };

var products = new List<Product>

{

new Product { Name = "Laptop", StockLevel = 10, Category = electronics },

new Product { Name = "Smartphone", StockLevel = 15, Category = electronics },

new Product { Name = "Bread", StockLevel = 30, Category = groceries },

new Product { Name = "Milk", StockLevel = 20, Category = groceries }

};

context.Categories.AddRange(electronics, groceries);

context.Products.AddRange(products);

context.SaveChanges();

}

var inventory = context.Products.Include(p => p.Category).ToList();

Console.WriteLine("Retail Inventory:");

foreach (var product in inventory)

{

Console.WriteLine($"- {product.Name} ({product.Category.Name}) - Stock: {product.StockLevel}");

}

}

**Output:**

