SupersetID: 6362191

Lab 1: Understanding ORM with a Retail Inventory System

Codes:

Category.cs :

namespace RetailInventory.Models;

public class Category

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public List<Product> Products { get; set; } = new();

}

Product.cs :

namespace RetailInventory.Models;

public class Product

{

    public int Id { get; set; }

     public string Name { get; set; } = string.Empty;

    public int Stock { get; set; }

    public int CategoryId { get; set; }

    public Category? Category { get; set; }

}

AppDbContext.cs :

using Microsoft.EntityFrameworkCore;

using RetailInventory.Models;

namespace RetailInventory;

public class AppDbContext : DbContext

{

    public DbSet<Product> Products => Set<Product>();

    public DbSet<Category> Categories => Set<Category>();

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

        optionsBuilder.UseSqlServer("Server=localhost\\SQLEXPRESS;Database=RetailDB;Trusted\_Connection=True;TrustServerCertificate=True;");

    }

}

Program.cs :

using RetailInventory;

using RetailInventory.Models;

using Microsoft.EntityFrameworkCore;

using var db = new AppDbContext();

if (!db.Categories.Any())

{

    var electronics = new Category

    {

        Name = "Electronics",

        Products = new List<Product>

        {

            new Product { Name = "Laptop", Stock = 10 },

            new Product { Name = "Smartphone", Stock = 25 },

            new Product { Name = "Headphones", Stock = 15 }

        }

    };

    var groceries = new Category

    {

        Name = "Groceries",

        Products = new List<Product>

        {

            new Product { Name = "Apples", Stock = 100 },

            new Product { Name = "Milk", Stock = 40 },

            new Product { Name = "Bread", Stock = 60 }

        }

    };

     var clothing = new Category

    {

        Name = "Clothing",

        Products = new List<Product>

        {

            new Product { Name = "T-Shirt", Stock = 30 },

            new Product { Name = "Jeans", Stock = 20 },

            new Product { Name = "Jacket", Stock = 10 }

        }

    };

    db.Categories.AddRange(electronics, groceries, clothing);

    db.SaveChanges();

}

var products = db.Products.Include(p => p.Category).ToList();

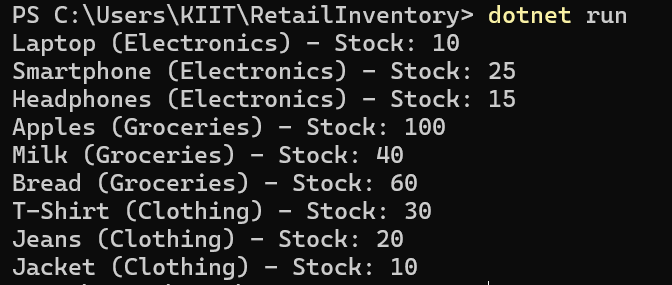
foreach (var product in products)

{

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

}

Output:



Lab 2: Setting Up the Database Context for a Retail Store

Codes :

Category.cs :

namespace RetailStore.Models;

public class Category

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public List<Product> Products { get; set; } = new();

}

Product.cs :

namespace RetailStore.Models;

public class Product

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

    public Category? Category { get; set; }

}

AppDbContextt.cs :

using Microsoft.EntityFrameworkCore;

using RetailStore.Models;

namespace RetailStore;

public class AppDbContext : DbContext

{

    public DbSet<Product> Products => Set<Product>();

    public DbSet<Category> Categories => Set<Category>();

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

        optionsBuilder.UseSqlServer("Server=localhost\\SQLEXPRESS;Database=RetailDB;Trusted\_Connection=True;TrustServerCertificate=True;");

    }

}

Program.cs :

using RetailStore;

using RetailStore.Models;

using Microsoft.EntityFrameworkCore;

using var db = new AppDbContext();

if (!db.Categories.Any())

{

    var category = new Category

    {

        Name = "Electronics",

        Products = new List<Product>

        {

            new Product { Name = "Smartphone", Price = 699 },

            new Product { Name = "Laptop", Price = 1299 }

        }

    };

    db.Categories.Add(category);

    db.SaveChanges();

}

var products = db.Products.Include(p => p.Category).ToList();

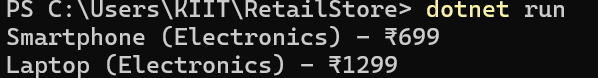
foreach (var product in products)

{

    Console.WriteLine($"{product.Name} ({product.Category?.Name}) - ₹{product.Price}");

}

Output:



Lab 3: Using EF Core CLI to Create and Apply Migrations

Codes :

Category.cs :

namespace RetailStore.Models;

public class Category

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public List<Product> Products { get; set; } = new();

}

Product.cs :

namespace RetailStore.Models;

public class Product

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

    public Category? Category { get; set; }

}

AppDbContext.cs :

using Microsoft.EntityFrameworkCore;

using RetailStore.Models;

namespace RetailStore;

public class AppDbContext : DbContext

{

    public DbSet<Product> Products => Set<Product>();

    public DbSet<Category> Categories => Set<Category>();

     protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

        optionsBuilder.UseSqlServer("Server=localhost\\SQLEXPRESS;Database=RetailDB;Trusted\_Connection=True;TrustServerCertificate=True;");

    }

}

Program.cs :

using RetailStore;

using RetailStore.Models;

using Microsoft.EntityFrameworkCore;

using var db = new AppDbContext();

if (!db.Categories.Any())

{

    var cat = new Category

    {

        Name = "Electronics",

        Products = new List<Product>

        {

            new Product { Name = "Laptop", Price = 1200 },

            new Product { Name = "Phone", Price = 600 }

        }

    };

    db.Categories.Add(cat);

    db.SaveChanges();

}

var products = db.Products.Include(p => p.Category).ToList();

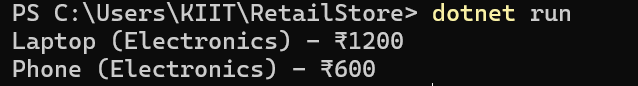
foreach (var product in products)

{

    Console.WriteLine($"{product.Name} ({product.Category?.Name}) - ₹{product.Price}");

}

Output :



Lab 4: Inserting Initial Data into the Database

Codes :

Category.cs :

namespace RetailStore.Models;

public class Category

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public List<Product> Products { get; set; } = new();

}

Product.cs :

namespace RetailStore.Models;

public class Product

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

    public Category? Category { get; set; }

}

AppDbContext.cs :

using Microsoft.EntityFrameworkCore;

using RetailStore.Models;

namespace RetailStore;

public class AppDbContext : DbContext

{

    public DbSet<Product> Products => Set<Product>();

    public DbSet<Category> Categories => Set<Category>();

     protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

        optionsBuilder.UseSqlServer("Server=localhost\\SQLEXPRESS;Database=RetailDB;Trusted\_Connection=True;TrustServerCertificate=True;");

    }

}

Program.cs :

using EFCoreProductApp.Models;

await InsertInitialDataAsync();

async Task InsertInitialDataAsync()

{

    using var context = new AppDbContext();

    await context.Database.EnsureCreatedAsync();

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

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

    await context.Categories.AddRangeAsync(electronics, groceries);

    var product1 = new Product { Name = "Laptop", Price = 75000, Category = electronics };

    var product2 = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };

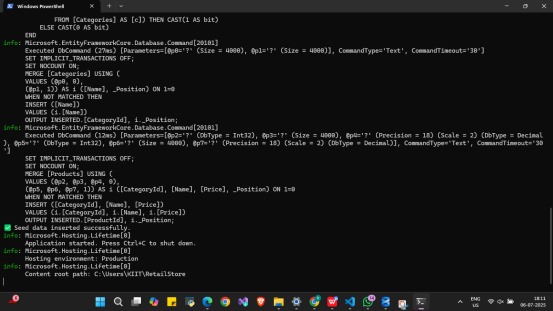
    await context.Products.AddRangeAsync(product1, product2);

    await context.SaveChangesAsync();

     Console.WriteLine("✅ Initial data inserted successfully.");

}

Output:



Lab 5: Retrieving Data from the Database

Codes :

Category.cs :

namespace RetailStore.Models;

public class Category

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public List<Product> Products { get; set; } = new();

}

Product.cs :

namespace RetailStore.Models;

public class Product

{

    public int Id { get; set; }

    public string Name { get; set; } = string.Empty;

    public decimal Price { get; set; }

    public int CategoryId { get; set; }

    public Category? Category { get; set; }

}

AppDbContext.cs :

using Microsoft.EntityFrameworkCore;

using RetailStore.Models;

namespace RetailStore;

public class AppDbContext : DbContext

{

    public DbSet<Product> Products => Set<Product>();

    public DbSet<Category> Categories => Set<Category>();

     protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

    {

        optionsBuilder.UseSqlServer("Server=localhost\\SQLEXPRESS;Database=RetailDB;Trusted\_Connection=True;TrustServerCertificate=True;");

    }

}

Program.cs :

using EFCoreProductApp.Models;

using Microsoft.EntityFrameworkCore;

class Program

{

    static async Task Main(string[] args)

    {

        using var context = new AppDbContext();

        Console.WriteLine("=== Lab 5: Retrieving Product Data ===");

        var products = await context.Products.ToListAsync();

        Console.WriteLine("\nAll Products:");

        foreach (var p in products)

        {

            Console.WriteLine($"{p.Name} - ₹{p.Price}");

        }

        var product = await context.Products.FindAsync(1);

        Console.WriteLine($"\nFound by ID 1: {product?.Name ?? "Not Found"}");

        var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);

        Console.WriteLine($"\nFirst Product Over ₹50000: {expensive?.Name ?? "None"}");

        Console.WriteLine("\nDone.");

    }

}

Output:

