Tên: Phạm Dương Minh Nhật

Mã sinh viên: 19IT182

# **Lab 1 & 3**

## **Program.cs**

using DemoDatabase.Models;

using System;

using System.Linq;

using Microsoft.EntityFrameworkCore;

using Org.BouncyCastle.Crypto.Tls;

namespace DemoDatabase

{

internal class Program

{

static void Main(string[] args)

{

mystoreContext mystore = new mystoreContext();

var products = from p in mystore.Products select new { p.ProductName, p.CategoryId };

foreach (var product in products)

{

Console.WriteLine($"Product Name: {product.ProductName}, Category ID: {product.CategoryId}");

}

Console.WriteLine("-xxx-xxx-");

IQueryable<Category> cats = mystore.Categories.Include(c => c.Products);

foreach (Category c in cats)

{

Console.WriteLine($"Category ID: {c.CategoryId} has {c.Products.Count} products.");

}

FilteredIncludes();

QueryingProducts();

AggregateProduct();

Console.ReadLine();

}

static void FilteredIncludes()

{

using var db = new mystoreContext();

Console.WriteLine("Enter Minimum for Units in Stock: ");

string unitInStock = Console.ReadLine();

int stock = int.Parse(unitInStock);

IQueryable<Category> cats = db.Categories.Include(c => c.Products.Where(p => p.UnitslnStock >= stock));

foreach (Category c in cats)

{

Console.WriteLine($"Category ID: {c.CategoryId} has {c.Products.Count} products.");

foreach(Product p in c.Products)

{

Console.WriteLine($"xxxx->: {p.ProductName} has {p.UnitslnStock} units in stock.");

}

}

}

static void QueryingProducts()

{

using (var context = new mystoreContext())

{

Console.WriteLine("Products that cost more than a price, the highest at the top");

string input;

double price;

do

{

Console.Write("Enter a product's price:");

input = Console.ReadLine();

}

while (!double.TryParse(input, out price));

IQueryable<Product> prod = context.Products

.Where(product => product.UnitPrice > price)

.OrderByDescending(product => product.UnitPrice);

Console.WriteLine("");

foreach (Product item in prod)

{

Console.WriteLine($"Produce: {item.ProductName} cost {item.UnitPrice} and has {item.UnitslnStock} left in stock");

}

}

}

static void AggregateProduct()

{

using (var context = new mystoreContext())

{

Console.WriteLine($"Product count: {context.Products.Count()}");

Console.WriteLine($"Highest product price: {context.Products.Max(p => p.UnitPrice)}");

Console.WriteLine($"Sum of unit in stock: {context.Products.Sum(p => p.UnitslnStock)}" );

Console.WriteLine($"Average Unit Price: {context.Products.Average(p => p.UnitslnStock)}" );

}

}

}

}

## **Product.cs**

using System;

using System.Collections.Generic;

#nullable disable

namespace DemoDatabase.Models

{

public partial class Product

{

public int ProductId { get; set; }

public string ProductName { get; set; }

public double UnitPrice { get; set; }

public int UnitslnStock { get; set; }

public int CategoryId { get; set; }

public virtual Category Category { get; set; }

}

}

## **mystoreContext.cs**

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

#nullable disable

namespace DemoDatabase.Models

{

public partial class mystoreContext : DbContext

{

public mystoreContext()

{

}

public mystoreContext(DbContextOptions<mystoreContext> options)

: base(options)

{

}

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

#warning To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.

optionsBuilder.UseMySQL("server=localhost;port=3306;user=root;password=;database=mystore");

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Category>(entity =>

{

entity.ToTable("categories");

entity.Property(e => e.CategoryId)

.HasColumnType("int(11)")

.HasColumnName("CategoryID");

entity.Property(e => e.CategoryName)

.IsRequired()

.HasMaxLength(40);

});

modelBuilder.Entity<Product>(entity =>

{

entity.ToTable("products");

entity.HasIndex(e => e.CategoryId, "Key");

entity.Property(e => e.ProductId)

.HasColumnType("int(11)")

.HasColumnName("ProductID");

entity.Property(e => e.CategoryId)

.HasColumnType("int(11)")

.HasColumnName("CategoryID");

entity.Property(e => e.ProductName)

.IsRequired()

.HasMaxLength(40);

entity.Property(e => e.UnitslnStock).HasColumnType("int(11)");

entity.HasOne(d => d.Category)

.WithMany(p => p.Products)

.HasForeignKey(d => d.CategoryId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("Key");

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

}

## **Category.cs**

using System;

using System.Collections.Generic;

#nullable disable

namespace DemoDatabase.Models

{

public partial class Category

{

public Category()

{

Products = new HashSet<Product>();

}

public int CategoryId { get; set; }

public string CategoryName { get; set; }

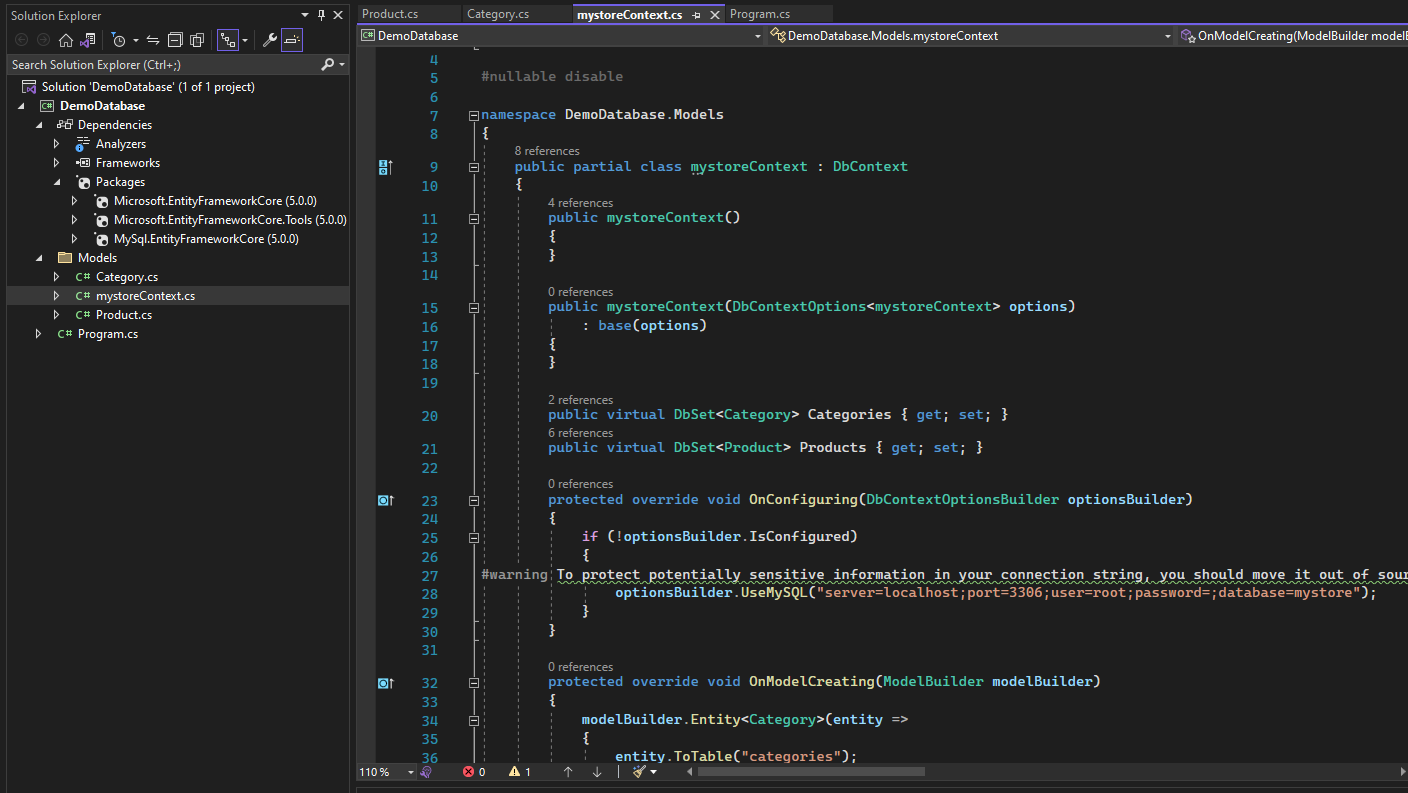
public virtual ICollection<Product> Products { get; set; }

}

}

Text

Description automatically generated



# **Lab 2**

## **Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ManageCategoriesApp2

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.SetHighDpiMode(HighDpiMode.SystemAware);

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

## **MyStockDBContext.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.Extensions.Configuration;

namespace ManageCategoriesApp2

{

public class Category

{

public Category() { }

[Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int CategoryId { get; set; }

public string CategoryName { get; set; }

}

internal class MyStockDBContext:DbContext

{

public MyStockDBContext() { }

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

var builder = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json", optional: true, reloadOnChange: true);

IConfigurationRoot configuration= builder.Build();

optionsBuilder.UseMySQL(configuration.GetConnectionString("MyStockDB"));

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Category>()

.Property(category => category.CategoryName)

.IsRequired()

.HasMaxLength(40);

modelBuilder.Entity<Category>().HasData(

new Category { CategoryId = 1, CategoryName = "Name 1" },

new Category { CategoryId = 2, CategoryName = "Name 2" },

new Category { CategoryId = 3, CategoryName = "Name 3" });

}

}

}

## **ManageCategories.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ManageCategoriesApp2

{

public sealed class ManageCategories

{

private static ManageCategories instance = null;

private static readonly object instanceLock = new object();

private ManageCategories() { }

public static ManageCategories Instance

{

get

{

lock (instanceLock)

{

if (instance == null)

{

instance = new ManageCategories();

}

return instance;

}

}

}

public List<Category> GetCategories()

{

List<Category> categories;

try

{

using MyStockDBContext stock = new MyStockDBContext();

categories = stock.Catagories.ToList();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

return categories;

}

public void InsertCategory(Category category)

{

try

{

using MyStockDBContext stock = new MyStockDBContext();

stock.Catagories.Add(category);

stock.SaveChanges();

}

catch(Exception ex)

{

throw new Exception(ex.Message);

}

}

public void UpdateCategory(Category category)

{

try

{

using MyStockDBContext stock = new MyStockDBContext();

stock.Entry<Category>(category).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

stock.SaveChanges();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

public void DeleteCategory(Category category)

{

try

{

using MyStockDBContext stock = new MyStockDBContext();

var cate = stock.Catagories.SingleOrDefault(c=> c.CategoryId == category.CategoryId);

stock.Catagories.Remove(cate);

stock.SaveChanges();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

}

}

## **20221116130714\_Initial.cs**

using Microsoft.EntityFrameworkCore.Migrations;

using MySql.EntityFrameworkCore.Metadata;

namespace ManageCategoriesApp2.Migrations

{

public partial class Initial : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Catagories",

columns: table => new

{

CategoryId = table.Column<int>(type: "int", nullable: false)

.Annotation("MySQL:ValueGenerationStrategy", MySQLValueGenerationStrategy.IdentityColumn),

CategoryName = table.Column<string>(type: "varchar(40)", maxLength: 40, nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Catagories", x => x.CategoryId);

});

migrationBuilder.InsertData(

table: "Catagories",

columns: new[] { "CategoryId", "CategoryName" },

values: new object[] { 1, "Name 1" });

migrationBuilder.InsertData(

table: "Catagories",

columns: new[] { "CategoryId", "CategoryName" },

values: new object[] { 2, "Name 2" });

migrationBuilder.InsertData(

table: "Catagories",

columns: new[] { "CategoryId", "CategoryName" },

values: new object[] { 3, "Name 3" });

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Catagories");

}

}

}

Graphical user interface

Description automatically generated