Lê Hữu Hoàn

Home work 27/10

Lập trình c# 3

A screenshot of a computer program

Description automatically generated

CODE:

|  |
| --- |
| // See https://aka.ms/new-console-template for more information  using DTO;  using DAO;  List<Book> list = await new BookDAO().All();  foreach (Book j in list) {  Console.WriteLine($"{j.Name},{j.BookId}");  }  bool i = await new BookDAO().UpdateOneRecord(new Book {  BookId = 3,  Name = "Test",  });  Console.WriteLine($"Update: {i}");  bool c = await new BookDAO().InsertOneRecordAsync(new Book {  Publication = "27/03/2022",  Name = "Test 1",  });  Console.WriteLine($"Insert: {c}");  List<Book> lists = await new BookDAO().All();  foreach (Book k in lists) {  Console.WriteLine($"{k.Name},{k.BookId}");  }  bool d = await new BookDAO().DeleteOneRecord(2);  Console.WriteLine($"Delete: {d}");  List<Book> lists1 = await new BookDAO().All();  foreach (Book kz in lists1) {  Console.WriteLine($"{kz.Name},{kz.BookId}");  } |

|  |
| --- |
| using DTO;  using Microsoft.EntityFrameworkCore;  using Models;  namespace DAO {  public class BookDAO {  private BooksContext ctx = new BooksContext();  public async Task<List<Book>> All() {  try {  return await ctx.books.ToListAsync();  } catch {  throw;  }  }  public async Task<List<Book>> Find(string keywork) {  try {  return await ctx.books.Where(s => EF.Functions.Like(s.Name, $"%{keywork}%")).ToListAsync();  } catch {  throw;  }  }  public async Task<bool> InsertOneRecordAsync(Book book) {  try {  await ctx.books.AddAsync(book);  int result = await ctx.SaveChangesAsync();  if (result >= 1) {  return true;  } else {  return false;  }  } catch {  throw;  }  }  public async Task<bool> UpdateOneRecord(Book book) {  try {  var product = await ctx.books.Where(s => s.BookId == book.BookId).FirstAsync();  if (product != null) {  product.Name = book.Name;  await ctx.SaveChangesAsync();  }  return true;  } catch {  throw;  }  }  public async Task<bool> DeleteOneRecord(int id) {  try {  var product = await ctx.books.Where(s => s.BookId == id).FirstAsync();  if (product != null) {  ctx.books.Remove(product);  await ctx.SaveChangesAsync();  }  return true;  } catch {  throw;  }  }  }  } |

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel.DataAnnotations.Schema;  using System.ComponentModel.DataAnnotations;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace DTO {  [Table("Books")]  public class Book {  [Key]  public int BookId { set; get; }  [Required]  [StringLength(50)]  public string Name { set; get; }  [StringLength(50)]  public string Publication { set; get; }  }  } |

|  |
| --- |
| using DTO;  using Microsoft.EntityFrameworkCore;  namespace Models {  public class BooksContext : DbContext {  public DbSet<Book> books { set; get; }  private const string connect = "Server=localhost;Database=Student;User ID=sa;Password=123456;Trusted\_Connection=True;TrustServerCertificate=True;";  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {  base.OnConfiguring(optionsBuilder);  optionsBuilder.UseSqlServer(connect);  }  public async Task CreateDatabase() {  String databasename = Database.GetDbConnection().Database;  Console.WriteLine("Tạo " + databasename);  bool result = await Database.EnsureCreatedAsync();  string resultstring = result ? "Success" : "database arealy exits";  Console.WriteLine($"CSDL {databasename} : {resultstring}");  }  public async Task DeleteDatabase() {  String databasename = Database.GetDbConnection().Database;  bool deleted = await Database.EnsureDeletedAsync();  string deletionInfo = deleted ? "đã xóa" : "không xóa được";  Console.WriteLine($"{databasename} {deletionInfo}");  }  }  } |