using Microsoft.EntityFrameworkCore; using Project_2.Models; var builder = WebApplication.CreateBuilder(args); // Add services to the container. builder.Services.AddDbContext(options => options.UseSqlServer(builder.Configuration.GetConnectionString("EFConStr"))); builder.Services.AddControllersWithViews(); var app = builder.Build(); // Configure the HTTP request pipeline. if (!app.Environment.IsDevelopment()) { app.UseExceptionHandler("/Home/Error"); } app.UseStaticFiles(); app.UseRouting(); app.UseAuthorization(); app.MapControllerRoute(name: "default", pattern: "{controller=Home}/{action=Index}/{id?}"); app.Run(); { "Logging": { "LogLevel": { "Default": "Information", "Microsoft.AspNetCore": "Warning" } }, "ConnectionStrings": { "EFConStr": "server=DESKTOP-U064AL2;database=Ecommerce;trusted_connection=true;" }, "AllowedHosts": "*" }

Project_2

- Home
- Privacy
- Orders
- Products

@RenderBody()
© 2024 - Project_2 - Privacy
@await RenderSectionAsync("Scripts", required: false) using System; using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; using Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using Microsoft.EntityFrameworkCore; using Project_2.Models; namespace Project_2.Controllers { public class ProductsController : Controller { private readonly EcommerceContext _context; public ProductsController(EcommerceContext context) { _context = context; } // GET: Products public async Task Index() { var ecommerceContext = _context.Products.Include(p => p.OidNavigation); return View(await ecommerceContext.ToListAsync()); } // GET: Products/Details/5 public async Task Details(int? id) { if (id == null || _context.Products == null) { return NotFound(); } var product = await _context.Products .Include(p => p.OidNavigation) .FirstOrDefaultAsync(m => m.Pid == id); if (product == null) { return NotFound(); } return View(product); } // GET: Products/Create public IActionResult Create() { ViewData["Oid"] = new SelectList(_context.Orders, "Oid", "Oid"); return View(); } // POST: Products/Create // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task Create([Bind("Pid,Pname,Pprice,Oid")] Product product) { if (ModelState.IsValid) { _context.Add(product); await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); } ViewData["Oid"] = new SelectList(_context.Orders, "Oid", "Oid", product.Oid); return View(product); } // GET: Products/Edit/5 public async Task Edit(int? id) { if (id == null || _context.Products == null) { return NotFound(); } var product = await _context.Products.FindAsync(id); if (product == null) { return NotFound(); } ViewData["Oid"] = new SelectList(_context.Orders, "Oid", "Oid", product.Oid); return View(product); } // POST: Products/Edit/5 // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task Edit(int id, [Bind("Pid,Pname,Pprice,Oid")] Product product) { if (id != product.Pid) { return NotFound(); } if (ModelState.IsValid) { try { _context.Update(product); await _context.SaveChangesAsync(); } catch (DbUpdateConcurrencyException) { if (!ProductExists(product.Pid)) { return NotFound(); } else { throw; } } return RedirectToAction(nameof(Index)); } ViewData["Oid"] = new SelectList(_context.Orders, "Oid", "Oid", product.Oid); return View(product); } // GET: Products/Delete/5 public async Task Delete(int? id) { if (id == null || _context.Products == null) { return NotFound(); } var product = await _context.Products .Include(p => p.OidNavigation) .FirstOrDefaultAsync(m => m.Pid == id); if (product == null) { return NotFound(); } return View(product); } // POST: Products/Delete/5 [HttpPost, ActionName("Delete")] [ValidateAntiForgeryToken] public async Task DeleteConfirmed(int id) { if (_context.Products == null) { return Problem("Entity set 'EcommerceContext.Products' is null."); } var product = await _context.Products.FindAsync(id); if (product != null) { _context.Products.Remove(product); } await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); } private bool ProductExists(int id) { return (_context.Products?.Any(e => e.Pid == id)).GetValueOrDefault(); } } } using System; using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; using Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.Rendering; using Microsoft.EntityFrameworkCore; using Project_2.Models; namespace Project_2.Controllers { public class OrdersController : Controller { private readonly EcommerceContext _context; public OrdersController(EcommerceContext context) { _context = context; } // GET: Orders public async Task Index() { return _context.Orders != null ? View(await _context.Orders.ToListAsync()) : Problem("Entity set 'EcommerceContext.Orders' is null."); } // GET: Orders/Details/5 public async Task Details(int? id) { if (id == null || _context.Orders == null) { return NotFound(); } var order = await _context.Orders .FirstOrDefaultAsync(m => m.Oid == id); if (order == null) { return NotFound(); } return View(order); } // GET: Orders/Create public IActionResult Create() { return View(); } // POST: Orders/Create // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task Create([Bind("Oid,Odate,Oaddress")] Order order) { if (ModelState.IsValid) { _context.Add(order); await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); } return View(order); } // GET: Orders/Edit/5 public async Task Edit(int? id) { if (id == null || _context.Orders == null) { return NotFound(); } var order = await _context.Orders.FindAsync(id); if (order == null) { return NotFound(); } return View(order); } // POST: Orders/Edit/5 // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task Edit(int id, [Bind("Oid,Odate,Oaddress")] Order order) { if (id != order.Oid) { return NotFound(); } if (ModelState.IsValid) { try { _context.Update(order); await _context.SaveChangesAsync(); } catch (DbUpdateConcurrencyException) { if (!OrderExists(order.Oid)) { return NotFound(); } else { throw; } } return RedirectToAction(nameof(Index)); } return View(order); } // GET: Orders/Delete/5 public async Task Delete(int? id) { if (id == null || _context.Orders == null) { return NotFound(); } var order = await _context.Orders .FirstOrDefaultAsync(m => m.Oid == id); if (order == null) { return NotFound(); } return View(order); } // POST: Orders/Delete/5 [HttpPost, ActionName("Delete")] [ValidateAntiForgeryToken] public async Task DeleteConfirmed(int id) { if (_context.Orders == null) { return Problem("Entity set 'EcommerceContext.Orders' is null."); } var order = await _context.Orders.FindAsync(id); if (order != null) { _context.Orders.Remove(order); } await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); } private bool OrderExists(int id) { return (_context.Orders?.Any(e => e.Oid == id)).GetValueOrDefault(); } } } using System; using System.Collections.Generic; using Microsoft.EntityFrameworkCore; using Microsoft.EntityFrameworkCore.Metadata; namespace Project_2.Models { public partial class EcommerceContext : DbContext { public EcommerceContext() { } public EcommerceContext(DbContextOptions options) : base(options) { } public virtual DbSet Orders { get; set; } = null!; public virtual DbSet Products { get; set; } = null!; 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.UseSqlServer("server=DESKTOP-U064AL2;database=Ecommerce;trusted_connection=true;"); } } protected override void OnModelCreating(ModelBuilder modelBuilder) { modelBuilder.Entity(entity => { entity.HasKey(e => e.Oid) .HasName("PK__Orders__CB3E4F310A85B3B0"); entity.Property(e => e.Oid).ValueGeneratedNever(); entity.Property(e => e.Oaddress) .HasMaxLength(40) .IsUnicode(false); entity.Property(e => e.Odate).HasColumnType("date"); }); modelBuilder.Entity(entity => { entity.HasKey(e => e.Pid) .HasName("PK__Products__C5705938B69966BE"); entity.Property(e => e.Pid).ValueGeneratedNever(); entity.Property(e => e.Pname) .HasMaxLength(20) .IsUnicode(false); entity.HasOne(d => d.OidNavigation) .WithMany(p => p.Products) .HasForeignKey(d => d.Oid) .HasConstraintName("FK__Products__Oid__4BAC3F29"); }); OnModelCreatingPartial(modelBuilder); } partial void OnModelCreatingPartial(ModelBuilder modelBuilder); } }