// ------------------------

// ✅ INSTRUCTIONS:

// ------------------------

// 1. Create a new WPF .NET Framework project in Visual Studio 2022

// 2. Add Entity Framework 6 via NuGet: Install-Package EntityFramework

// 3. Add DevExpress WPF 24.1 controls

// 4. Add your Northwind EDMX: Right-click project > Add > New Item > Data > ADO.NET Entity Data Model

// - Choose EF Designer from Database

// - Connect to Northwind

// - Select only the "Orders" table

// - Name it NorthwindModel.edmx

// 5. Add folders: Models, ViewModels, Views

// 6. Place each class below in the appropriate folder

// ------------------------

// --------------------------------------

// 🧩 Models/OrderRepository.cs

// --------------------------------------

using System.Collections.ObjectModel;

using System.Data.Entity;

using System.Linq;

namespace YourApp.Models

{

public class OrderRepository

{

private NorthwindEntities \_context;

public ObservableCollection<Order> Orders { get; private set; }

public OrderRepository()

{

\_context = new NorthwindEntities();

\_context.Orders.Load();

Orders = \_context.Orders.Local;

}

public void SaveChanges() => \_context.SaveChanges();

public void AddOrder(Order order) => \_context.Orders.Add(order);

public void RemoveOrder(Order order) => \_context.Orders.Remove(order);

public void Reload() => \_context = new NorthwindEntities();

}

}

// --------------------------------------

// 🧠 ViewModels/OrdersViewModel.cs

// --------------------------------------

using DevExpress.Mvvm;

using System.Collections.ObjectModel;

using YourApp.Models;

namespace YourApp.ViewModels

{

public class OrdersViewModel : ViewModelBase

{

private readonly OrderRepository \_repo;

public ObservableCollection<Order> Orders => \_repo.Orders;

public OrdersViewModel()

{

\_repo = new OrderRepository();

}

public void SaveAllChanges() => \_repo.SaveChanges();

public void AddOrder(Order order) => \_repo.AddOrder(order);

public void DeleteOrder(Order order) => \_repo.RemoveOrder(order);

public void ResetChanges()

{

\_repo.Reload();

}

}

}

// --------------------------------------

// 🖼️ Views/OrdersView.xaml

// --------------------------------------

<dx:ThemedWindow x:Class="YourApp.Views.OrdersView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:dx="http://schemas.devexpress.com/winfx/2008/xaml/core"

xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"

xmlns:vm="clr-namespace:YourApp.ViewModels"

Title="Orders" Width="1000" Height="600">

<dx:ThemedWindow.DataContext>

<vm:OrdersViewModel/>

</dx:ThemedWindow.DataContext>

<DockPanel>

<StackPanel DockPanel.Dock="Top" Orientation="Horizontal" Margin="5">

<Button Content="Save Changes" Padding="5" Margin="5" Click="OnSaveClick"/>

<Button Content="Undo Changes" Padding="5" Margin="5" Click="OnUndoClick"/>

</StackPanel>

<dxg:GridControl Name="grid" ItemsSource="{Binding Orders}" AutoGenerateColumns="AddNew">

<dxg:GridControl.View>

<dxg:TableView CellDoubleClick="OnCellDoubleClick" AutoWidth="True" ShowGroupPanel="False"/>

</dxg:GridControl.View>

</dxg:GridControl>

</DockPanel>

</dx:ThemedWindow>

// --------------------------------------

// Views/OrdersView.xaml.cs

// --------------------------------------

using System.Windows;

using YourApp.ViewModels;

namespace YourApp.Views

{

public partial class OrdersView : DevExpress.Xpf.Core.ThemedWindow

{

public OrdersView() => InitializeComponent();

private void OnSaveClick(object sender, RoutedEventArgs e)

{

if (DataContext is OrdersViewModel vm)

vm.SaveAllChanges();

}

private void OnUndoClick(object sender, RoutedEventArgs e)

{

if (DataContext is OrdersViewModel vm)

vm.ResetChanges();

MessageBox.Show("Changes have been reset.", "Reset", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void OnCellDoubleClick(object sender, DevExpress.Xpf.Grid.CellClickEventArgs e)

{

if (DataContext is OrdersViewModel vm && e.Row is Order order)

{

var editor = new EditOrderWindow(order);

editor.ShowDialog();

}

}

}

}

// --------------------------------------

// 🪟 Views/EditOrderWindow.xaml

// --------------------------------------

<Window x:Class="YourApp.Views.EditOrderWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Edit Order" Height="300" Width="400">

<Grid Margin="10">

<StackPanel>

<StackPanel Orientation="Horizontal" Margin="0,5">

<TextBlock Text="Customer ID" Width="100"/>

<TextBox x:Name="txtCustomerID" Width="200" Text="{Binding CustomerID, UpdateSourceTrigger=PropertyChanged, ValidatesOnDataErrors=True, NotifyOnValidationError=True}" />

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0,5">

<TextBlock Text="Order Date" Width="100"/>

<DatePicker x:Name="dpOrderDate" Width="200" SelectedDate="{Binding OrderDate, UpdateSourceTrigger=PropertyChanged, ValidatesOnDataErrors=True, NotifyOnValidationError=True}" />

</StackPanel>

<TextBlock Foreground="Red" x:Name="errorBlock" Visibility="Collapsed" Text="Please fill in all fields."/>

<Button Content="Done" Margin="10" HorizontalAlignment="Right" Click="DoneClick"/>

</StackPanel>

</Grid>

</Window>

// --------------------------------------

// Views/EditOrderWindow.xaml.cs

// --------------------------------------

using System;

using System.Windows;

namespace YourApp.Views

{

public partial class EditOrderWindow : Window

{

public EditOrderWindow(Order order)

{

InitializeComponent();

DataContext = order;

}

private void DoneClick(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(txtCustomerID.Text) || !dpOrderDate.SelectedDate.HasValue)

{

errorBlock.Visibility = Visibility.Visible;

return;

}

errorBlock.Visibility = Visibility.Collapsed;

DialogResult = true;

Close();

}

}

}