Tarefa 3.4 Programming 3

Matheus Santos dos Anjos

Explicação:

No padrão MVVM, a responsabilidade é separada entre Model (lógica de negócios e dados), View (interface gráfica) e ViewModel (intermediário entre Model e View). Essa divisão facilita o teste, manutenção e escalabilidade da aplicação.

A vinculação de dados (data binding) permite que a View seja automaticamente atualizada quando o ViewModel muda. Isso elimina a necessidade de código explícito para atualização da interface, reduzindo erros e promovendo desacoplamento entre a lógica da aplicação e a UI.

Exemplo 1: MVVM com C#

Neste exemplo, vamos criar um aplicativo simples que exibe o nome de um usuário e permite que ele seja alterado através de um `TextBox`. O valor é atualizado na `View` automaticamente quando alterado no `ViewModel` usando **vinculação de dados**.

1. Model

O Model contém apenas dados ou lógica de negócios. Neste caso, o Model é simples e contém uma propriedade `Nome`.

```
public class UserModel
{
    public string Nome { get; set; }
}
```

2. ViewModel

O ViewModel é o responsável por fornecer os dados para a View. Ele expõe as propriedades de forma que a View possa ser vinculada a elas.

```
using System.ComponentModel;

public class UserViewModel : INotifyPropertyChanged
{
    private UserModel _user;
    public event PropertyChangedEventHandler PropertyChanged;

    public UserViewModel()
    {
        _user = new UserModel() { Nome = "Robert Sanchez" };
    }

    public string Nome
    {
        get { return _user.Nome; }
        set
        {
            if (_user.Nome != value)
            {
                _user.Nome = value;
                OnPropertyChanged("Nome");
            }
        }
    }

    protected virtual void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

3. View (XAML)

Aqui, a interface de usuário é criada. O `TextBox` está vinculado à propriedade `Nome` do ViewModel.

```
<Window x:Class="MVVMExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MVVM Example" Height="200" Width="300">
    <Grid>
        <TextBox Text="{Binding Nome}" VerticalAlignment="Center" HorizontalAlignment="Center" Width="200"/>
    </Grid>
</Window>

```

4. Code-Behind (MainWindow.xaml.cs)

Aqui, a View (`MainWindow`) é configurada para usar o ViewModel e estabelecer a vinculação de dados.

```
using System.Windows;

namespace MVVMExample
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            this.DataContext = new UserViewModel();
        }
    }
}
```

Exemplo 2: Delegates e Eventos em C#

Neste exemplo, vamos criar um programa simples com **delegates** e **eventos** para gerenciar a interação de um botão que exibe uma mensagem quando clicado.

1. Delegate e Evento

```
using System;

public class Button
{
    // Define o delegate para o evento
    public delegate void ClickEventHandler(object sender, EventArgs e);

    // Evento que será disparado quando o botão for clicado
    public event ClickEventHandler Click;

    // Método que dispara o evento
    public void OnClick()
    {
        Click?.Invoke(this, EventArgs.Empty);
    }
}
```

2. Handler de Evento (Classe Principal)

Aqui, vamos definir um método que será chamado quando o evento for disparado.

```
using System;

public class Program
{
    // Método que será chamado ao clicar no botão
    public static void Button_Click(object sender, EventArgs e)
    {
        Console.WriteLine("Botão foi clicado!");
    }

    public static void Main()
    {
        Button button = new Button();

        // Associando o evento Click ao método Button_Click
        button.Click += Button_Click;

        // Simulando um clique no botão
        button.OnClick();

        Console.ReadLine();
    }
}
```

Explicação do Exemplo:

- **Delegates e Eventos**: O delegate ClickEventHandler define a assinatura dos métodos que podem ser associados ao evento Click. O evento é disparado pela chamada de OnClick().
- **Vinculação de Dados em MVVM**: No exemplo de MVVM, a vinculação de dados (Text="{Binding Nome}") garante que, sempre que a propriedade Nome mudar no ViewModel, a View seja automaticamente atualizada, refletindo essa mudança sem necessidade de código explícito de manipulação de eventos.