<UserControl x:Class="ChessUI.Welcome" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:local="clr-namespace:ChessUI" mc:Ignorable="d" d:DesignHeight="450" d:DesignWidth="800"> <Border Width="300" Height="300" Style="{StaticResource BorderStyle}"> <Grid> <Grid.RowDefinitions> <RowDefinition Height="\*"/> <RowDefinition Height="auto"/> <RowDefinition Height="1.2\*"/> </Grid.RowDefinitions> <Grid.ColumnDefinitions> <ColumnDefinition Width="\*"/> <ColumnDefinition Width="\*"/> <ColumnDefinition Width="\*"/> </Grid.ColumnDefinitions> <TextBlock Grid.ColumnSpan="3" Text="SELECT MODE" Style="{StaticResource TextStyle}" FontSize="28"/> <Button Grid.Row="2" Grid.Column="0" Content="QUIT" Style="{StaticResource ButtonStyle}" Width="80" Height="30" Command="{Binding QuitCommand}" x:Name="Quit" Click="Quit\_Click"/> <Button Grid.Row="2" Grid.Column="2" Content="PLAY" Style="{StaticResource ButtonStyle}" Width="80" Height="30" Command="{Binding PlayCommand}"/> <ListBox Grid.Row="1" Grid.ColumnSpan="3" Style="{StaticResource ListBoxStyle}" Width="250"> <ListBoxItem Width="100" HorizontalAlignment="Center" VerticalAlignment="Center">Bot Mode</ListBoxItem> <ListBoxItem Width="120" HorizontalAlignment="Center" VerticalAlignment="Center">Human Mode</ListBoxItem> </ListBox> </Grid> </Border> </UserControl> using System.Windows; using System.Windows.Controls; namespace ChessUI { /// <summary> /// Interaction logic for Welcome.xaml /// </summary> public partial class Welcome : UserControl { public enum WelcomeOption { NewGame, BotGame, Quit } public event Action<WelcomeOption> OptionSelected; public Welcome() { InitializeComponent(); } private void Quit\_Click(object sender, RoutedEventArgs e) { OptionSelected?.Invoke(WelcomeOption.Quit); } private void NewGame\_Click(object sender, RoutedEventArgs e) { OptionSelected?.Invoke(WelcomeOption.NewGame); } private void BotGame\_Click(object sender, RoutedEventArgs e) { OptionSelected?.Invoke(WelcomeOption.BotGame); } } } ans in vietnamese

**ChatGPT đã nói:**

Dưới đây là bản giải thích bằng tiếng Việt cho mã XAML và C# mà bạn cung cấp:

**🧩 Mô tả tệp XAML (Welcome.xaml)**

Đây là giao diện người dùng của một màn hình chào mừng (Welcome screen) cho ứng dụng chơi cờ (ChessUI):

* UserControl này có một **khung (Border)** kích thước 300x300 và dùng các style được định nghĩa sẵn như BorderStyle, ButtonStyle, TextStyle, v.v.
* Giao diện được chia làm 3 hàng (RowDefinition) và 3 cột (ColumnDefinition).
* Thành phần bên trong:
  + **TextBlock** hiển thị dòng chữ "SELECT MODE".
  + **ListBox** chứa 2 tùy chọn: "Bot Mode" và "Human Mode".
  + **Hai nút bấm (Button)**:
    - Nút "QUIT" ở cột 0, gọi hàm Quit\_Click khi bấm.
    - Nút "PLAY" ở cột 2, gọi command PlayCommand.

**🔧 Mô tả mã C# (Welcome.xaml.cs)**

Lớp Welcome kế thừa từ UserControl chứa logic xử lý sự kiện:

**👇 Các thành phần chính:**

* **Enum WelcomeOption**:
  + Dùng để xác định hành động mà người dùng chọn: NewGame, BotGame, Quit.
* **Sự kiện OptionSelected**:
  + Là một sự kiện public kiểu Action<WelcomeOption>, dùng để thông báo cho phần khác của chương trình biết người dùng đã chọn gì.

**🧠 Các hàm sự kiện:**

* Quit\_Click: Khi người dùng nhấn nút "QUIT", chương trình gọi OptionSelected?.Invoke(WelcomeOption.Quit).
* NewGame\_Click và BotGame\_Click: Có sẵn nhưng không được gán trong XAML hiện tại.

❗ **Ghi chú**: Nút "PLAY" hiện tại **không gọi được** NewGame\_Click hoặc BotGame\_Click, vì không có xử lý Click và không xác định được người dùng chọn chế độ nào trong ListBox. Bạn cần bổ sung logic để xác định chế độ đã chọn và gọi sự kiện tương ứng.

**✅ Gợi ý chỉnh sửa để hoàn thiện chức năng "PLAY":**

**Thêm x:Name="ModeListBox" cho ListBox:**

xml

Sao chépChỉnh sửa

<ListBox x:Name="ModeListBox" Grid.Row="1"

Grid.ColumnSpan="3"

Style="{StaticResource ListBoxStyle}" Width="250">

**Thêm Click handler cho nút PLAY:**

xml

Sao chépChỉnh sửa

<Button Grid.Row="2"

Grid.Column="2"

Content="PLAY"

Style="{StaticResource ButtonStyle}"

Width="80"

Height="30"

Click="Play\_Click"/>

**Thêm hàm Play\_Click trong Welcome.xaml.cs:**

csharp

Sao chépChỉnh sửa

private void Play\_Click(object sender, RoutedEventArgs e)

{

if (ModeListBox.SelectedItem is ListBoxItem selectedItem)

{

string selectedText = selectedItem.Content.ToString();

if (selectedText == "Bot Mode")

OptionSelected?.Invoke(WelcomeOption.BotGame);

else if (selectedText == "Human Mode")

OptionSelected?.Invoke(WelcomeOption.NewGame);

}

}

Nếu bạn cần, mình có thể giúp bạn tạo luôn logic cho việc xử lý khởi động ván cờ theo chế độ được chọn.

**Bạn đã nói:**

đây là 1 usercontrol được gọi trong mainwindow ở mainwindow xaml, toi define nó dưới dạng contentcontrol u feel me toi muốn usercontrol này hiển thị và toi chọn được các select đó (listviewitem) u feel me ?

**ChatGPT đã nói:**

Yes, I feel you. Bạn đang sử dụng UserControl (Welcome) như một phần trong MainWindow, được hiển thị thông qua ContentControl. Bạn muốn:

* Welcome.xaml hiển thị đúng trong MainWindow.
* Người dùng có thể chọn 1 trong 2 chế độ (Bot Mode / Human Mode).
* Nút PLAY phải thực sự lấy thông tin từ ListBox và gửi tín hiệu chọn chế độ về MainWindow.

**✅ Giải pháp đầy đủ**

**1. Trong MainWindow.xaml**

Bạn định nghĩa:

xml

Sao chépChỉnh sửa

<ContentControl x:Name="MainContent"/>

**2. Trong MainWindow.xaml.cs**

Bạn cần nạp Welcome vào MainContent, rồi xử lý OptionSelected.

csharp

Sao chépChỉnh sửa

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

LoadWelcomeScreen();

}

private void LoadWelcomeScreen()

{

var welcomeControl = new Welcome();

welcomeControl.OptionSelected += WelcomeControl\_OptionSelected;

MainContent.Content = welcomeControl;

}

private void WelcomeControl\_OptionSelected(Welcome.WelcomeOption option)

{

switch (option)

{

case Welcome.WelcomeOption.NewGame:

MessageBox.Show("Người chơi chọn: Human Mode");

break;

case Welcome.WelcomeOption.BotGame:

MessageBox.Show("Người chơi chọn: Bot Mode");

break;

case Welcome.WelcomeOption.Quit:

Application.Current.Shutdown();

break;

}

}

}

**3. Sửa lại Welcome.xaml để chọn ListBoxItem hoạt động**

xml

Sao chépChỉnh sửa

<ListBox x:Name="ModeListBox"

Grid.Row="1"

Grid.ColumnSpan="3"

Style="{StaticResource ListBoxStyle}" Width="250"

SelectedIndex="0">

<ListBoxItem>Bot Mode</ListBoxItem>

<ListBoxItem>Human Mode</ListBoxItem>

</ListBox>

**4. Xử lý nút PLAY trong Welcome.xaml.cs**

csharp

Sao chépChỉnh sửa

private void Play\_Click(object sender, RoutedEventArgs e)

{

if (ModeListBox.SelectedItem is ListBoxItem selectedItem)

{

string selectedMode = selectedItem.Content.ToString();

if (selectedMode == "Bot Mode")

OptionSelected?.Invoke(WelcomeOption.BotGame);

else if (selectedMode == "Human Mode")

OptionSelected?.Invoke(WelcomeOption.NewGame);

}

}

**✅ Tổng kết**

Bạn đã có:

* UserControl Welcome có sự kiện báo cho MainWindow.
* MainWindow nhận sự kiện và chuyển trạng thái theo lựa chọn.
* PLAY sẽ dựa vào ListBox để xác định chế độ.

Nếu bạn cần sau khi chọn chế độ thì thay đổi toàn bộ MainContent thành bàn cờ (chẳng hạn), mình cũng có thể hỗ trợ đoạn đó nhé.

Let me know if you want the "next step."

**Bạn đã nói:**

WelcomeControl\_OptionSelected o dau ra ?

**ChatGPT đã nói:**

Great question. Câu hỏi của bạn là:

“WelcomeControl\_OptionSelected ở đâu ra?”

**✅ Trả lời ngắn gọn:**

WelcomeControl\_OptionSelected là một **hàm xử lý sự kiện** (event handler) mà **bạn tự định nghĩa** trong MainWindow.xaml.cs.

Nó được **gắn vào sự kiện OptionSelected của Welcome** bằng dòng:

csharp

Sao chépChỉnh sửa

welcomeControl.OptionSelected += WelcomeControl\_OptionSelected;

**✅ Vậy hàm này phải viết thế nào?**

Bạn **phải tự tạo** hàm đó như sau trong MainWindow.xaml.cs:

csharp

Sao chépChỉnh sửa

private void WelcomeControl\_OptionSelected(Welcome.WelcomeOption option)

{

switch (option)

{

case Welcome.WelcomeOption.NewGame:

MessageBox.Show("Người chơi chọn: Human Mode");

break;

case Welcome.WelcomeOption.BotGame:

MessageBox.Show("Người chơi chọn: Bot Mode");

break;

case Welcome.WelcomeOption.Quit:

Application.Current.Shutdown();

break;

}

}

**🧠 Tổng quát hoá:**

* OptionSelected là một sự kiện event Action<WelcomeOption> bên trong Welcome.xaml.cs.
* MainWindow cần đăng ký lắng nghe sự kiện này thông qua +=.
* WelcomeControl\_OptionSelected là nơi bạn xử lý khi người dùng chọn chế độ từ UserControl.

Nếu bạn muốn làm phần này động hơn, ví dụ như **sau khi chọn PLAY thì hiển thị bàn cờ**, mình có thể hướng dẫn tiếp nhé.

**Bạn đã nói:**

co cach khac khong

**ChatGPT đã nói:**

Có chứ! Nếu bạn **không muốn dùng event (OptionSelected) và handler (WelcomeControl\_OptionSelected)** theo kiểu “code-behind” truyền thống, bạn có vài **cách thay thế hiện đại và gọn gàng hơn**, tùy vào phong cách bạn chọn:

**✅ Cách 1: Dùng MVVM – Binding + Command**

Nếu bạn đang theo mô hình MVVM, bạn **không cần dùng event thủ công**, mà xử lý mọi thứ qua **Command** và **ViewModel**.

**Ưu điểm:**

* Rõ ràng, sạch sẽ, dễ test.
* Không cần truy cập UserControl từ MainWindow kiểu trực tiếp.

**✅ Cách 2: Truy cập trực tiếp từ MainWindow thông qua ContentControl.Content**

Nếu bạn chỉ muốn nhanh gọn, có thể **ép kiểu MainContent.Content về Welcome** và đọc trực tiếp thuộc tính hoặc gọi phương thức:

**Trong MainWindow.xaml.cs:**

csharp

Sao chépChỉnh sửa

private void PlayButton\_OnClick(object sender, RoutedEventArgs e)

{

if (MainContent.Content is Welcome welcomeControl)

{

var selectedItem = welcomeControl.ModeListBox.SelectedItem as ListBoxItem;

string mode = selectedItem?.Content?.ToString();

if (mode == "Bot Mode")

{

MessageBox.Show("Chế độ Bot");

}

else if (mode == "Human Mode")

{

MessageBox.Show("Chế độ Người");

}

}

}

📝 Lưu ý: ModeListBox phải được đặt là public hoặc bạn tạo 1 phương thức public trong Welcome để lấy giá trị chế độ hiện tại.

**Ví dụ thêm phương thức:**

Trong Welcome.xaml.cs:

csharp

Sao chépChỉnh sửa

public string GetSelectedMode()

{

if (ModeListBox.SelectedItem is ListBoxItem item)

return item.Content.ToString();

return null;

}

Trong MainWindow:

csharp

Sao chépChỉnh sửa

string mode = welcomeControl.GetSelectedMode();

**✅ Cách 3: Dùng Delegate/Callback từ MainWindow gán vào Welcome**

Thay vì sự kiện OptionSelected, bạn truyền vào 1 delegate:

csharp

Sao chépChỉnh sửa

public Action<string> OnPlaySelected;

Trong Welcome.xaml.cs, khi PLAY được nhấn:

csharp

Sao chépChỉnh sửa

private void Play\_Click(object sender, RoutedEventArgs e)

{

if (ModeListBox.SelectedItem is ListBoxItem item)

{

OnPlaySelected?.Invoke(item.Content.ToString());

}

}

Rồi trong MainWindow:

csharp

Sao chépChỉnh sửa

var welcome = new Welcome();

welcome.OnPlaySelected = mode =>

{

MessageBox.Show($"Người dùng chọn: {mode}");

};

MainContent.Content = welcome;

**✅ Tóm tắt:**

| **Cách làm** | **Đặc điểm** |
| --- | --- |
| **Event truyền thống** | An toàn, tách biệt, dễ hiểu – bạn đang dùng cách này rồi |
| **MVVM Command + Binding** | Gọn hơn, chuẩn kiến trúc MVVM nếu dùng ViewModel |
| **Truy cập trực tiếp** | Đơn giản, nhanh nhưng coupling cao, phù hợp cho demo/POC |
| **Dùng Delegate Callback** | Rất linh hoạt, không cần khai báo event, dùng cho truyền dữ liệu ngắn gọn |

Nếu bạn nói rõ hơn: *"tôi đang không muốn dùng event vì..."* thì mình có thể tư vấn cách phù hợp nhất luôn nhé.

**Bạn đã nói:**

nen dung cach nao ?

**ChatGPT đã nói:**

Câu hỏi rất hay và quan trọng: **Nên dùng cách nào?**  
Câu trả lời phụ thuộc vào **quy mô dự án, phong cách lập trình, và nhu cầu mở rộng** của bạn.

**✅ Đề xuất ngắn gọn:**

| **Nếu bạn đang...** | **Thì nên chọn...** |
| --- | --- |
| Viết app nhỏ, nhanh, thử nghiệm | ❗ **Cách 2 – Truy cập trực tiếp** |
| Muốn code gọn mà không cần MVVM | ✅ **Cách 3 – Dùng delegate** |
| Dự án thật, có thể mở rộng, chuyên nghiệp | 🔥 **Cách 1 – MVVM + Command** |

**📌 Giải thích chi tiết**

**🔹 Cách 1 – MVVM + Command (Khuyến nghị cho dự án thật)**

* **Ưu điểm**:
  + Rất rõ ràng, dễ maintain, chia lớp tốt (View <-> ViewModel).
  + Dễ viết Unit Test.
  + Có thể thay thế giao diện mà không ảnh hưởng logic.
* **Nhược điểm**:
  + Phải set up thêm ViewModel, Binding, ICommand.

**=> Dành cho app bài bản, quy mô trung bình trở lên.**

**🔹 Cách 2 – Truy cập trực tiếp từ MainContent.Content**

* **Ưu điểm**:
  + Đơn giản, dễ hiểu.
  + Không cần dùng delegate, event.
* **Nhược điểm**:
  + Coupling cao: MainWindow phụ thuộc cụ thể vào Welcome.
  + Khó mở rộng / bảo trì.

**=> Dành cho app demo, cá nhân, nhỏ.**

**🔹 Cách 3 – Delegate callback (tốt cho app nhỏ/giữa)**

* **Ưu điểm**:
  + Gọn, dễ kiểm soát hơn event.
  + Không cần Binding, không cần ICommand.
  + Welcome không phụ thuộc vào MainWindow.
* **Nhược điểm**:
  + Nếu truyền nhiều callback, dễ rối nếu thiếu tổ chức.

**=> Dành cho app vừa, hoặc bạn chưa muốn dùng MVVM.**

**💬 Lời khuyên cuối**

* Nếu bạn đang **học WPF** → Dùng **event hoặc delegate** để hiểu flow logic.
* Nếu bạn đang **làm app thực tế** → Học và dùng **MVVM + ICommand**.
* Nếu bạn chỉ muốn xong nhanh → Dùng cách **truy cập trực tiếp** cũng ổn, nhưng nên cải tiến sau.

Nếu bạn nói rõ hơn app của bạn đang ở mức nào (demo, bài tập, dự án thật, làm việc nhóm...), mình có thể khuyên cách tốt nhất luôn nhé.

**Bạn đã nói:**

vay dung cach

**ChatGPT đã nói:**

Rồi, bạn đang muốn mình **chọn giúp cách nên dùng**, và bạn hỏi rất thực tế.

**✅ Kết luận: Dùng cách 3 – Delegate Callback là hợp lý nhất cho bạn hiện tại**

**✔ Vì sao?**

* Gọn gàng hơn event (ít boilerplate hơn).
* Không cần học MVVM ngay bây giờ.
* Không bị rối như việc truy cập trực tiếp từ ContentControl.
* Dễ maintain và dễ nâng cấp sang MVVM sau này nếu cần.

**🛠 Hướng dẫn nhanh dùng delegate callback**

**1. \*\*Trong**

**Bạn đã nói:**

vay dung cach 1

**ChatGPT đã nói:**

Tuyệt! Nếu bạn muốn **dùng cách 1 – MVVM + Command**, thì mình sẽ hướng dẫn bạn từng bước ngắn gọn, dễ làm, đúng chuẩn.

**✅ Mục tiêu:**

* Welcome.xaml là một UserControl.
* Dùng **ICommand** để xử lý nút PLAY và QUIT.
* Dữ liệu được chọn từ ListBox sẽ được **ràng buộc (Binding)** vào ViewModel.
* Không cần event hoặc delegate thủ công.

**🧱 Bước 1: Tạo ViewModel**

Tạo một lớp mới: WelcomeViewModel.cs

csharp

Sao chépChỉnh sửa

using System;

using System.ComponentModel;

using System.Windows.Input;

namespace ChessUI

{

public class WelcomeViewModel : INotifyPropertyChanged

{

private string \_selectedMode;

public string SelectedMode

{

get => \_selectedMode;

set

{

\_selectedMode = value;

OnPropertyChanged(nameof(SelectedMode));

}

}

public ICommand PlayCommand { get; }

public ICommand QuitCommand { get; }

public event Action<string> PlayModeSelected;

public event Action QuitRequested;

public WelcomeViewModel()

{

PlayCommand = new RelayCommand(ExecutePlay, CanExecutePlay);

QuitCommand = new RelayCommand(\_ => QuitRequested?.Invoke());

}

private bool CanExecutePlay(object arg) => !string.IsNullOrEmpty(SelectedMode);

private void ExecutePlay(object obj)

{

PlayModeSelected?.Invoke(SelectedMode);

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string name)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

}

}

}

🧠 RelayCommand là một lớp nhỏ để hiện thực ICommand. Xem bước 2.

**🧱 Bước 2: Tạo lớp RelayCommand.cs (nếu chưa có)**

csharp

Sao chépChỉnh sửa

using System;

using System.Windows.Input;

public class RelayCommand : ICommand

{

private readonly Action<object> \_execute;

private readonly Func<object, bool> \_canExecute;

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object parameter) => \_canExecute?.Invoke(parameter) ?? true;

public void Execute(object parameter) => \_execute(parameter);

public event EventHandler CanExecuteChanged

{

add => CommandManager.RequerySuggested += value;

remove => CommandManager.RequerySuggested -= value;

}

}

**🧱 Bước 3: Ràng buộc trong Welcome.xaml.cs**

Trong Welcome.xaml.cs, sửa lại constructor:

csharp

Sao chépChỉnh sửa

public partial class Welcome : UserControl

{

public WelcomeViewModel ViewModel { get; } = new WelcomeViewModel();

public Welcome()

{

InitializeComponent();

DataContext = ViewModel;

}

}

**🧱 Bước 4: Ràng buộc XAML – Welcome.xaml**

Sửa ListBox và các nút như sau:

xml

Sao chépChỉnh sửa

<ListBox x:Name="ModeListBox"

Grid.Row="1"

Grid.ColumnSpan="3"

SelectedValue="{Binding SelectedMode, Mode=TwoWay}"

SelectedValuePath="Content"

Style="{StaticResource ListBoxStyle}" Width="250">

<ListBoxItem Content="Bot Mode"/>

<ListBoxItem Content="Human Mode"/>

</ListBox>

<Button Grid.Row="2"

Grid.Column="0"

Content="QUIT"

Style="{StaticResource ButtonStyle}"

Width="80"

Height="30"

Command="{Binding QuitCommand}" />

<Button Grid.Row="2"

Grid.Column="2"

Content="PLAY"

Style="{StaticResource ButtonStyle}"

Width="80"

Height="30"

Command="{Binding PlayCommand}" />

**🧱 Bước 5: Sử dụng trong MainWindow.xaml.cs**

csharp

Sao chépChỉnh sửa

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

var welcome = new Welcome();

var vm = welcome.ViewModel;

vm.PlayModeSelected += mode =>

{

MessageBox.Show($"Người dùng chọn: {mode}");

// TODO: Chuyển sang màn hình chơi game tương ứng

};

vm.QuitRequested += () =>

{

Application.Current.Shutdown();

};

MainContent.Content = welcome;

}

}

**✅ Kết quả:**

* Giao diện kết nối ViewModel qua binding.
* Không cần viết Click="...".
* Tách biệt hoàn toàn UI và logic.
* Mở rộng dễ dàng (ví dụ thêm các chế độ khác, ghi log, test...).

Nếu bạn muốn mình **tạo sẵn toàn bộ template MVVM cho ChessUI**, mình có thể gửi toàn bộ project cấu trúc mẫu. Bạn muốn không?

**Bạn đã nói:**

co

**ChatGPT đã nói:**

Tuyệt! Dưới đây là một **template đầy đủ theo mô hình MVVM** cho phần Welcome của ứng dụng **ChessUI**, bạn có thể dùng trực tiếp hoặc mở rộng thêm tùy nhu cầu.

**🗂 Cấu trúc thư mục gợi ý:**

Sao chépChỉnh sửa

ChessUI/

├── Views/

│ └── Welcome.xaml

│ └── Welcome.xaml.cs

├── ViewModels/

│ └── WelcomeViewModel.cs

├── Commands/

│ └── RelayCommand.cs

├── MainWindow.xaml

├── MainWindow.xaml.cs

**🔧 1. RelayCommand.cs (trong Commands/)**

csharp

Sao chépChỉnh sửa

using System;

using System.Windows.Input;

namespace ChessUI.Commands

{

public class RelayCommand : ICommand

{

private readonly Action<object> \_execute;

private readonly Func<object, bool> \_canExecute;

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object parameter) => \_canExecute?.Invoke(parameter) ?? true;

public void Execute(object parameter) => \_execute(parameter);

public event EventHandler CanExecuteChanged

{

add => CommandManager.RequerySuggested += value;

remove => CommandManager.RequerySuggested -= value;

}

}

}

**🧠 2. WelcomeViewModel.cs (trong ViewModels/)**

csharp

Sao chépChỉnh sửa

using System;

using System.ComponentModel;

using System.Windows.Input;

using ChessUI.Commands;

namespace ChessUI.ViewModels

{

public class WelcomeViewModel : INotifyPropertyChanged

{

private string \_selectedMode;

public string SelectedMode

{

get => \_selectedMode;

set

{

\_selectedMode = value;

OnPropertyChanged(nameof(SelectedMode));

}

}

public ICommand PlayCommand { get; }

public ICommand QuitCommand { get; }

public event Action<string> PlayModeSelected;

public event Action QuitRequested;

public WelcomeViewModel()

{

PlayCommand = new RelayCommand(\_ => PlayModeSelected?.Invoke(SelectedMode),

\_ => !string.IsNullOrWhiteSpace(SelectedMode));

QuitCommand = new RelayCommand(\_ => QuitRequested?.Invoke());

}

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string propertyName) =>

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

**🎨 3. Welcome.xaml (trong Views/)**

xml

Sao chépChỉnh sửa

<UserControl x:Class="ChessUI.Views.Welcome"

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

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

xmlns:local="clr-namespace:ChessUI.Views"

xmlns:viewModels="clr-namespace:ChessUI.ViewModels"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<UserControl.DataContext>

<viewModels:WelcomeViewModel />

</UserControl.DataContext>

<Border Width="300" Height="300" BorderBrush="Gray" BorderThickness="1" CornerRadius="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="1.2\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.ColumnSpan="3" Text="SELECT MODE" FontSize="28"

HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,20,0,20"/>

<ListBox Grid.Row="1" Grid.ColumnSpan="3"

SelectedValue="{Binding SelectedMode, Mode=TwoWay}"

SelectedValuePath="Content"

HorizontalAlignment="Center" Width="200" Margin="0,10,0,10">

<ListBoxItem Content="Bot Mode"/>

<ListBoxItem Content="Human Mode"/>

</ListBox>

<Button Grid.Row="2" Grid.Column="0" Content="QUIT" Width="80" Height="30"

Command="{Binding QuitCommand}"/>

<Button Grid.Row="2" Grid.Column="2" Content="PLAY" Width="80" Height="30"

Command="{Binding PlayCommand}"/>

</Grid>

</Border>

</UserControl>

**⚙️ 4. Welcome.xaml.cs (code-behind, rất đơn giản)**

csharp

Sao chépChỉnh sửa

using System.Windows.Controls;

using ChessUI.ViewModels;

namespace ChessUI.Views

{

public partial class Welcome : UserControl

{

public WelcomeViewModel ViewModel => (WelcomeViewModel)DataContext;

public Welcome()

{

InitializeComponent();

}

}

}

**🏠 5. MainWindow.xaml**

xml

Sao chépChỉnh sửa

<Window x:Class="ChessUI.MainWindow"

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

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

xmlns:views="clr-namespace:ChessUI.Views"

Title="Chess UI" Height="450" Width="800">

<Grid>

<ContentControl x:Name="MainContent" />

</Grid>

</Window>

**🧩 6. MainWindow.xaml.cs**

csharp

Sao chépChỉnh sửa

using System.Windows;

using ChessUI.Views;

using ChessUI.ViewModels;

namespace ChessUI

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

var welcomeView = new Welcome();

var vm = welcomeView.ViewModel;

vm.PlayModeSelected += mode =>

{

MessageBox.Show($"Người dùng chọn: {mode}");

// TODO: Replace MainContent with Game screen here

};

vm.QuitRequested += () =>

{

Application.Current.Shutdown();

};

MainContent.Content = welcomeView;

}

}

}

**✅ Tổng kết**

🎯 Với cấu trúc này, bạn có:

* Giao diện tách biệt hoàn toàn logic (ViewModel).
* Xử lý lệnh qua ICommand, không dùng Click.
* Có thể dễ dàng mở rộng sang các màn hình khác như GameBoard, Settings, v.v.

Nếu bạn muốn mình **nén thành một project .zip sẵn sàng chạy**, bạn có thể cung cấp email, hoặc dùng GitHub để mình hướng dẫn tạo repo demo.  
Hoặc, mình có thể giúp bạn tiếp tục xây dựng màn chơi cờ (Game View), bạn chỉ cần nói "let's go".