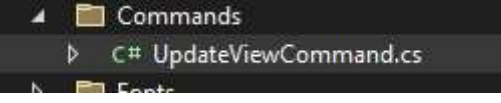


DOKUMENTACJA PROJEKTU

Kilka sprostowań wobec kodu aplikacji.

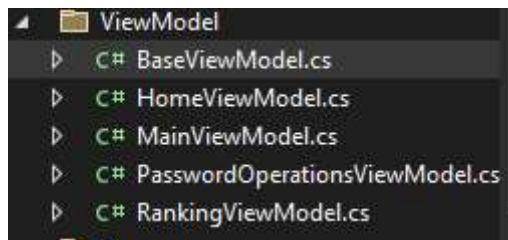
1. Interfejs ICommand



```
7 using System.Windows.Input;
8
9 namespace HangMen.Commands
10 {
11     Odwwołania: 2
12     public class UpdateViewCommand : ICommand
13     {
14         1 odwołanie
15         private MainViewModel viewModel;
16         public UpdateViewCommand(MainViewModel viewModel)
17         {
18             this.viewModel = viewModel;
19         }
20         public event EventHandler? CanExecuteChanged;
21
22         Odwwołania: 0
23         public bool CanExecute(object? parameter)
24         {
25             return true;
26         }
27
28         Odwwołania: 0
29         public void Execute(object? parameter)
30         {
31             if(parameter.ToString() == "Home")
32             {
33                 viewModel.SelectedViewModel = new HomeViewModel();
34             }
35             else if(parameter.ToString() == "Account")
36             {
37                 viewModel.SelectedViewModel = new RankingViewModel();
38             }
39             else if(parameter.ToString() == "PasswordOperations")
40             {
41                 viewModel.SelectedViewModel = new PasswordOperationsViewModel();
42             }
43         }
44     }
45 }
```

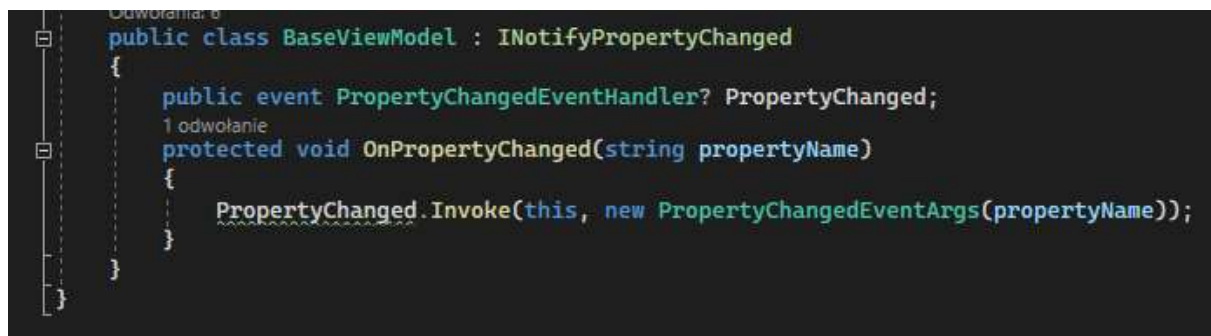
W znacznym skrócie interfejs ten służy (przynajmniej w mojej aplikacji) do sprawnego zarządzania i przełączania się między poszczególnymi viewModelami aplikacji.

viewModely umieszczone są w folderze viewModel:

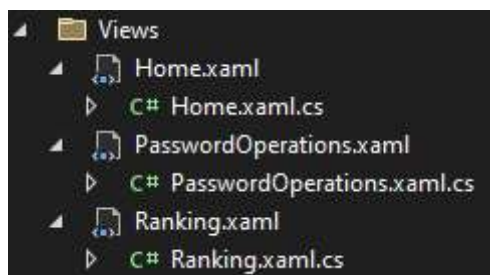


A o tym który viewModel ma w danym momencie być uruchomiony decyduje klasa

BaseViewModel.cs



a widoki które mają być uruchomione przez viewModel znajdują się w folderze Views:



więcej na ten temat można przeczytać tutaj

<https://learn.microsoft.com/pl-pl/dotnet/api/system.windows.input.icommand?view=net-7.0>

2. Problem z niektórymi messageBoxami

```
Application.Current.Dispatcher.BeginInvoke(DispatcherPriority.Normal, (Action)(() =>
{
    MessageBox.Show("Problem with DataBase connection!", "DB ERROR", MessageBoxButton.OK, MessageBoxImage.Warning);
}));
```

Gdzie nie gdzie w kodzie mojej aplikacji można spotkać się z takim zapisem messageBox'a jest to spowodowane tym że w mojej aplikacji można spotkać czasem metody asynchroniczne czyli na przykład metody których używamy przy tworzeniu timera wtedy trzeba otoczyć naszego messageBox'a tym o to zapisem aby działał on na wszystkich urządzeniach poprawnie

(u mnie w domu messageBox'y działały bez problemu lecz w szkole u pana w sali wywalały błędy dlatego chciałem się upewnić że wszystko będzie działać dobrze na każdym urządzeniu)

3. Timer rozgrywki

```
dispatcherTimer = new DispatcherTimer();
dispatcherTimer.Tick += addSecond;
dispatcherTimer.Interval = new TimeSpan(0, 0, 1);
```

Tutaj timer został utworzony na bazie klasy dispatcherTimer która pozwala na ustawienie funkcji wykonywanej po upływie czasu określonego w interwale

w 1 linii tworzona jest instancja klasy DispatcherTimer

w 2 linii ustawiany jest event który ma się wydarzyć po upływie interwału

w 3 linii ustawiana jest długość danego interwału

DispatcherTimer jest to odłam programowania asynchronicznego więc nie "freezuje" on aplikacji podczas jej działania co jest bardzo wielkim ułatwieniem