Delegate command leży w class ButtonViewAViewModel

dodaj:

using Microsoft.Practices.Prism.Mvvm;

using System.Windows.Input;

using System.Windows;

okazuje się, że używany przez nas delegate command implementuje interfejs ICommand

dzięki temu można zastąpić: na

public ICommand SwitchViewCommand { get; set; }

Wywołuje się ją w ButtonViewA.xaml

 <Grid>

<Button Content="Przełącz na widok A" Command="{Binding SwitchViewCommand}" />

</Grid>

Composite command

Zmieniam icommand na

public ComponentCommand SwitchViewCommand { get; set; }

nastepnie znajdz SwitchViewCommand = new DelegateCommand(SwitchView); I zamien na

SwitchViewCommand = new CompositeCommand();

SwitchViewCommand.RegisterCommand(new DelegateCommand(SwitchView));

Powinno działać

Tworzymy nową klasę w ModuleA:

public static class GlobalCommands

using Microsoft.Practices.Prism.Mvvm;

using Microsoft.Practices.Prism.Commands;

public static CompositeCommand SwitchViewCommand = new CompositeCommand();

GlobalCommands.SwitchViewCommand.RegisterCommand(new DelegateCommand(SwitchView));

spójrz do ButtonAView – powinno być:

 <Grid>

 <Button Content="Przełącz na widok A" Command="{Binding SwitchViewCommand}" />

</Grid>

Zamień na

 <Grid>

 <Button Content="Przełącz na widok A" Command="{x:Static local:GlobalCommands.SwitchViewCommand}" />

</Grid>

Pamiętaj żeby dodać:

xmlns:local="clr-namespace:ModuleA"

uzupełnij EditorCatalogViewModell.cs

 public ModuleAViewViewModel(IAuthority authority)

 {

 this.authority = authority;

 GlobalCommands.SwitchViewCommand.RegisterCommand(new DelegateCommand(kominukat));

 }

 public void kominukat()

 {

 MessageBox.Show("druga komenda");

 }

Przydatne usingi:

using Microsoft.Practices.Prism.Commands;

using System.Windows;

**IEventAggregator**

Dodaj nową klasę do modułu A == messageEvent zawierająca:

 public class messageEvent : CompositePresentationEvent<string> { }

dodaj do EditorCatalogViewModel

IEventAggregator \_eventAggregator;

Do konstruktora dodaj aggregator

 public EditorCatalogViewModel(IProductRepository productsRepository, IEventAggregator eventAggregator) : base(productsRepository)

I na początku:

\_eventAggregator = eventAggregator;

w add product dodaj:

\_eventAggregator.GetEvent<messageEvent>().Publish("dodano element");

W ModuleBViewViewModel dodaj

IEventAggregator \_eventAggregator;

W konstruktorze

\_eventAggregator = eventaggregator;

 eventaggregator.GetEvent<messageEvent>().Subscribe(komunikat,true);

I metoda:

public void komunikat(string tekst) { MessageBox.Show(“ otrzymano wiadomość od innego wątku:”+ tekst); }