Przewodnik krok po kroku:

## Aplikacja jest rozbudowywana. Jej podstawą jest: <https://msdn.microsoft.com/en-us/library/ff921174(v=pandp.40).aspx>

Do aplikacji dodajemy nowy moduł, który posłuży nam do dodawania nowych wydarzeń w osobnej karcie.

Na początu tworzymy widok byśmy wiedzieli co będziemy chcieli otrzymać(zaprogramować):

Tworzymy plik:

AddNewProjectView.xaml

Uzupełniamy go o poniższą treść:

<UserControl x:Class="UIComposition.EmployeeModule.Views.AddNewProjectView"

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

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

<Grid x:Name="LayoutRoot">

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#4e2c85" />

<GradientStop Color="#5f3d96" Offset="1" />

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Text="Nowy projekt" TextWrapping="Wrap" FontSize="18" Foreground="#eee" Margin="8" />

<Grid Grid.Row="1" Grid.Column="0" Width="Auto" Height="Auto" Margin="8">

<Grid.ColumnDefinitions>

<ColumnDefinition Width=".5\*" />

<ColumnDefinition Width=".5\*" />

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" Orientation="Vertical">

<TextBlock Text="Co zrobić:" Foreground="White" HorizontalAlignment="Left" Padding="0,5,5,5" FontWeight="Bold" />

<TextBox AutomationProperties.AutomationId="FirstNameTextBox" Height="Auto" Width="Auto" HorizontalAlignment="Stretch" Margin="0,5,100,5" Text="{Binding textBox}"/>

</StackPanel>

<StackPanel Grid.Column="1">

<TextBlock Text="Opis:" Foreground="White" HorizontalAlignment="Left" Padding="0,5,5,5" FontWeight="Bold" />

<TextBox AutomationProperties.AutomationId="LastNameTextBox" Height="Auto" Width="Auto" HorizontalAlignment="Stretch" Margin="0,5,100,5" Text="{Binding textBox1}"/>

<Button Content="Dodaj Zadanie" Foreground="White" Background="#333" Command="{Binding NewProjects}"/>

</StackPanel>

</Grid>

</Grid>

</UserControl>

Po dodaniu widoku potrzebujemy dodać w pliku cs widoku, w którym to obserwujemy zdarzenia:

using System.Windows.Controls;

using Microsoft.Practices.Prism.Regions;

using UIComposition.EmployeeModule.Models;

using UIComposition.EmployeeModule.ViewModels;

namespace UIComposition.EmployeeModule.Views

{

/// <summary>

/// Interaction logic for AddNewProject.xaml

/// </summary>

public partial class AddNewProjectView : UserControl

{

public AddNewProjectView(AddNewProjectViewModel addNewProjectViewModel)

{

InitializeComponent();

// Set the ViewModel as this View's data context.

this.DataContext = addNewProjectViewModel;

// This view is displayed in a region with a region context.

// The region context is defined as the currently selected employee

// When the region context is changed, we need to propogate the

// change to this view's view model.

RegionContext.GetObservableContext(this).PropertyChanged += (s, e)

=>

addNewProjectViewModel.CurrentEmployee =

RegionContext.GetObservableContext(this).Value

as Employee;

}

}

}

Po wykonaniu tych czynności przechodzimy do oprogramowania logiki aplikacji.

Tworzymy interfejs w którym implementujemy funkcję AddProjects

using UIComposition.EmployeeModule.Models;

namespace UIComposition.EmployeeModule.Services

{

/// <summary>

/// Data service interface.

/// </summary>

public interface IEmployeeDataService

{

Employees GetEmployees();

Projects GetProjects();

void AddProjects(string id, string zadanie, string opis);

}

}

Struktura takiego projektu wygląda następująco:

namespace UIComposition.EmployeeModule.Models

{

public class Project

{

public string Id { get; set; }

public string ProjectName { get; set; }

public string Role { get; set; }

}

}

Implementacja ViewModelu AddNewProject

Wiążemy akcje z GUI.

using System.ComponentModel;

using Microsoft.Practices.Prism;

using Microsoft.Practices.Prism.Commands;

using UIComposition.EmployeeModule.Models;

using UIComposition.EmployeeModule.Services;

namespace UIComposition.EmployeeModule.ViewModels

{

public class AddNewProjectViewModel : INotifyPropertyChanged

{

public string textBox {

get ;

set ;

}

public string textBox1

{

get;

set;

}

public AddNewProjectViewModel(IEmployeeDataService iEmployeeDataService)

{

NewProjects = new DelegateCommand(() =>

{

iEmployeeDataService.AddProjects(currentEmployee.Id,textBox,textBox1);

});

}

public DelegateCommand NewProjects

{

get ;

set ;

}

public string ViewName

{

get { return "Nowe zadanie"; }

}

private Employee currentEmployee;

public Employee CurrentEmployee

{

get { return this.currentEmployee; }

set

{

this.currentEmployee = value;

this.NotifyPropertyChanged("CurrentEmployee");

}

}

#region INotifyPropertyChanged Members

public event PropertyChangedEventHandler PropertyChanged;

private void NotifyPropertyChanged(string propertyName)

{

if (this.PropertyChanged != null)

{

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

}

}

#endregion

}

}

ViewModel deleguje nam akcje do Modelu.

Po zakończeniu poszczególnych kroków otrzymujemy nowy moduł odpowiedzialny za tworzenie nowych projektów.