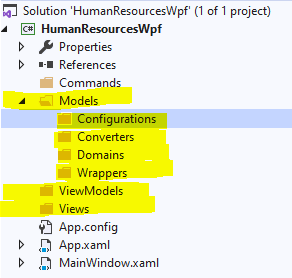
**Human Resources**

1 Tworzymy strukturę katalogów



2 Kopiujemy MainWindow.xaml do katalogu Views  
 w App.xaml modyfikujemy

|  |
| --- |
| <Application x:Class="HumanResourcesWpf.App"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:local="clr-namespace:HumanResourcesWpf"  StartupUri="Views\MainWindow.xaml">  <Application.Resources>    </Application.Resources>  </Application> |

3 Doinstalowujemy przez NuGet MahApps.Metro

Do App.xaml dodajemy

|  |
| --- |
| <Application x:Class="HumanResourcesWpf.App"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:local="clr-namespace:HumanResourcesWpf"  StartupUri="Views\MainWindow.xaml">    <Application.Resources>  <!--  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="Styles/ButtonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  --> |
| <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <!-- MahApps.Metro resource dictionaries. Make sure that all file names are Case Sensitive! -->  <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.xaml" />  <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Fonts.xaml" />  <!-- Theme setting -->  <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Themes/Light.Blue.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary> |
| </Application.Resources>    </Application> |

W MainWindow.xaml

|  |
| --- |
| <mah:MetroWindow x:Class="HumanResourcesWpf.Views.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:mah="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"  xmlns:i="http://schemas.microsoft.com/xaml/behaviors"  xmlns:local="clr-namespace:HumanResourcesWpf.Views"  mc:Ignorable="d"  Title="MainWindow" Height="450" Width="800">  <Grid>    </Grid>  </mah:MetroWindow> |

W MainWindow.xaml.cs

|  |
| --- |
| namespace HumanResourcesWpf.Views  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : MetroWindow  {  public MainWindow()  {  InitializeComponent();  }  }  } |

W katalogu ViewModel tworzymy model widoku ViewModels\MainWindowViewModel.cs  
oraz wgrywamy ViewModelBase.cs aby był dostęp do metody OnPropertChanged.

|  |
| --- |
| namespace HumanResources.ViewModels  {  class MainWindowViewModel : ViewModelBase  {  }  } |

W codebehind okna MainWindow (widoku) ustawiamy powiązanie z modelem widoku.

|  |
| --- |
| namespace HumanResources.Views  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : MetroWindow  {  public MainWindow()  {  InitializeComponent();  DataContext = new MainWindowViewModel();  }  }  } |

**Bindowanie zdarzeń – click() przycisków**

Po stronie XAML ( widoku )

|  |
| --- |
| <Button  Command ="{Binding ConectionConfigurationCommand}"  Content="Konfiguracja"  Height="30"  Width="100"  Margin="5"  /> |

Po stronie modelu widoku  
1) Tworzymy właściwość typu ICommand   
2) W konstruktorze pod tą właściwość podpinamy nowy obiekt typu RelayCommand  
3) Tworzymy metodę

|  |
| --- |
| namespace HumanResources.ViewModels  {  class MainWindowViewModel : ViewModelBase  {  public ICommand ConectionConfigurationCommand { get; set; }  public ICommand LoadedWindowCommand { get; set; }  public MainWindowViewModel()  {  ConectionConfigurationCommand = new RelayCommand(ConectionConfiguration);  LoadedWindowCommand = new RelayCommand(MainWindow\_Loaded);  }  private void MainWindow\_Loaded(object obj)  {  // TODO: MainWindow Loaded  return;  }  private void ConectionConfiguration(object obj)  {  var connectionConfigurationWindow = new ConnectionConfigurationView();  connectionConfigurationWindow.ShowDialog();  }  }  } |

**Bindowanie zdarzeń – inne niż click() przycisków**

Po stronie widoku XAML – dodajemy kod zapisany na żółto

|  |
| --- |
| <ComboBox  Height="30"  Width="150"  Margin="5"  ItemsSource="{Binding Groups}"  SelectedValue="{Binding SelectedGroupId}"  DisplayMemberPath="Name"  SelectedValuePath="Id">  <i:Interaction.Triggers>  <i:EventTrigger EventName="LostFocus">  <i:InvokeCommandAction Command="{Binding ComboBoxChanged}" />  </i:EventTrigger>  </i:Interaction.Triggers>  </ComboBox> |

Po stronie widoku modelu – 3 kroki, postępujemy dokładnie tak samo jak w poprzednim przypadku

|  |
| --- |
| class MainWindowViewModel : ViewModelBase  {  public ICommand ConectionConfigurationCommand { get; set; }  public ICommand LoadedWindowCommand { get; set; }  public ICommand ComboBoxChanged { get; set; }  public MainWindowViewModel()  {  ConectionConfigurationCommand = new RelayCommand(ConectionConfiguration);  LoadedWindowCommand = new RelayCommand(MainWindow\_Loaded);  ComboBoxChanged = new RelayCommand(ComboBox\_LostFocus);  }  private void ComboBox\_LostFocus(object obj)  {  // TODO: Oprogramowanie Lost Focus'a Combo Boxa z grupami  return;  }  private void MainWindow\_Loaded(object obj)  {  // TODO: MainWindow Loaded  return;  }  private void ConectionConfiguration(object obj)  {  var connectionConfigurationWindow = new ConnectionConfigurationView();  connectionConfigurationWindow.ShowDialog();  }  } |