Aplikacje tworzymy na podstawie aplikacji z poprzednich zajęć. Przerobimy ją aby wyświetlała listę produktów wraz z cenami w dwóch trybach. Do odczytu i do edycji.

Model oraz repozytorium

Model produktu: w ModuleA dodajemy plik Product.cs

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.Threading.Tasks;
6.
7.  namespace ModuleA.Models
8.  {
9.      public class Product
10.     {
11.         public long Id { get; set; }
12.         public string Name { get; set; }
13.         public decimal Price { get; set; }
14.
15.         public Product(long id, string name, decimal price)
16.         {
17.             this.Id = id;
18.             this.Name = name;
19.             this.Price = price;
20.         }
21.     }
22. }
```

Nasze produkty będą przechowywane w repozytorium. Tworzymy zatem interfejs IProductRepository.cs w tym samym module.

```
1.  using ModuleA.Models;
2.  using System;
3.  using System.Collections.Generic;
4.
5.  namespace ModuleA
6.  {
7.      public interface IProductRepository
8.      {
9.          void Add(Product product);
10.         void Update(Product product);
11.         IEnumerable<Product> GetAll();
12.     }
13. }
```

Na potrzeby projektu dodajemy klasę HardcodedProductRepository implementującą ten interfejs.

```
1.  using ModuleA.Models;
2.  using System;
3.  using System.Collections.Generic;
4.
5.  namespace ModuleA
6.  {
7.      public class HardcodedProductRepository : IProductRepository
8.      {
9.          private List<Product> products = new List<Product>()
10.         {
11.             new Product(0, "Apple", 0.25m),
12.             new Product(1, "Orange", 0.33m),
13.             new Product(2, "Strawberry", 0.10m),
14.         };
15.
16.         public void Add(Models.Product product)
17.         {
18.                         product.Id = products.Count;
19.             products.Add(product);
20.         }
21.
22.         public void Update(Models.Product product)
23.         {
24.             if (products != null)
25.                 products[(int)product.Id] = product;
26.         }
27.
28.         public IEnumerable<Models.Product> GetAll()
29.         {
30.             return products.ToArray();
31.         }
32.     }
33. }
```

Widoki i Modele Widoków

Widok do odczytu Catalog.xaml

```
<UserControl x:Class="ModuleA.Views.Catalog"
             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:prism="clr-
namespace:Microsoft.Practices.Prism.Mvvm;assembly=Microsoft.Practices.Prism.Mvvm.Desktop"
             prism:ViewModelLocator.AutoWireViewModel="True"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <ListView ItemsSource="{Binding Products}" Background="Azure">
        <ListView.ItemTemplate>
            <DataTemplate>
                <StackPanel Orientation="Horizontal">
                    <TextBlock Text="{Binding Name}"/>
                    <TextBlock Text="{Binding Price}"/>
                </StackPanel>
            </DataTemplate>
        </ListView.ItemTemplate>
    </ListView>
</UserControl>
```

Model widoku do odczytu CatalogViewModel.cs

```
using Microsoft.Practices.Prism.Mvvm;
using Microsoft.Practices.Prism.Regions;
using ModuleA.Models;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ModuleA.ViewModels
{
    public class CatalogViewModel : BindableBase, INavigationAware
    {
                private ObservableCollection<Product> products;
```

```
16.         public ObservableCollection<Product> Products
17.                 {
18.                         get
19.                         {
20.                                 return products;
21.                         }
22.                         protected set
23.                         {
24.                                 SetProperty<ObservableCollection<Product>>(ref products, val
    ue);
25.                         }
26.                 }
27.
28.         protected IProductRepository productsRepository;
29.
30.         public CatalogViewModel(IProductRepository productsRepository)
31.         {
32.             this.productsRepository = productsRepository;
33.             this.Products = new ObservableCollection<Product>(this.productsRepository.GetAll
    ());
34.         }
35.
36.                 public bool IsNavigationTarget(NavigationContext navigationContext)
37.                 {
38.                         return true;
39.                 }
40.
41.                 public void OnNavigatedFrom(NavigationContext navigationContext)
42.                 {
43.                 }
44.
45.                 public void OnNavigatedTo(NavigationContext navigationContext)
46.                 {
47.                         this.Products = new ObservableCollection<Product>(this.productsRepos
    itory.GetAll());
48.                 }
49.         }
50. }
```

Widok do edycji EditorCatalog.xaml

```
1.  <UserControl x:Class="ModuleA.Views.EditorCatalog"
2.              xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3.              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
4.              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5.              xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6.                          xmlns:prism="clr-
    namespace:Microsoft.Practices.Prism.Mvvm;assembly=Microsoft.Practices.Prism.Mvvm.Desktop"
7.              prism:ViewModelLocator.AutoWireViewModel="True"
8.              mc:Ignorable="d"
9.              d:DesignHeight="300" d:DesignWidth="300">
10.     <Grid>
11.                 <Grid.RowDefinitions>
12.                         <RowDefinition Height="1*"/>
13.                         <RowDefinition Height="5*"/>
14.                 </Grid.RowDefinitions>
15.                 <StackPanel Orientation="Horizontal" Grid.Row="0">
16.                         <Button Content="Add" Command="{Binding AddProductCommand}"/>
17.                         <Button Content="Update" Command="{Binding
    UpdateProductCommand}" CommandParameter="{Binding ElementName=ListV, Path=SelectedItem}"/>
18.                 </StackPanel>
19.                 <ListView ItemsSource="{Binding
    Products}" Background="LemonChiffon" Grid.Row="1" x:Name="ListV">
20.                         <ListView.ItemTemplate>
21.                                 <DataTemplate>
22.                                         <StackPanel Orientation="Horizontal">
23.                                                 <TextBox Text="{Binding Name}"/>
24.                                                 <TextBox Text="{Binding Price}"/>
25.                                         </StackPanel>
26.                                 </DataTemplate>
27.                         </ListView.ItemTemplate>
28.                 </ListView>
29.         </Grid>
30. </UserControl>
```

Model widoku do edycji EditorCatalogViewModel.cs

```
1.  using Microsoft.Practices.Prism.Commands;
2.  using Microsoft.Practices.Prism.Regions;
3.  using ModuleA.Models;
4.  using System;
5.  using System.Collections.Generic;
6.  using System.Collections.ObjectModel;
7.  using System.Linq;
8.  using System.Text;
9.  using System.Threading.Tasks;
10.
11. namespace ModuleA.ViewModels
```

```
{
        public class EditorCatalogViewModel : CatalogViewModel
        {
                public DelegateCommand AddProductCommand { get; set; }
                public DelegateCommand<Product> UpdateProductCommand { get; set; }

                public EditorCatalogViewModel(IProductRepository
    productsRepository) : base(productsRepository)
                {
                        AddProductCommand = new DelegateCommand(AddProduct);
                        UpdateProductCommand = new DelegateCommand<Product>(UpdateProduct);
                }

                private void AddProduct()
                {
                        var sampleProduct = new Product(0, "Name", 0.00m);
                        base.productsRepository.Add(sampleProduct);
                        base.Products.Add(sampleProduct);
                }

                private void UpdateProduct(Product product)
                {
                        if(product != null)
                                base.productsRepository.Update(product);
                }
        }
}
```

Usługa

Potrzebujemy jeszcze usługi do zarządzania czy jesteśmy w trybie do odczytu czy edycji. W tym celu tworzymy nowy projekt o nazwie Services i dodajemy w nim interfejs IAuthority.cs

```
namespace Services
{
    public interface IAuthority
    {
        bool IsEditor { get; set; }
    }
}
```

Oraz klasę która go implementuje SettingsAuthority.cs

```
1.  using System;
2.
3.  namespace Services
4.  {
5.      public class SettingsAuthority : IAuthority
6.      {
7.          public bool IsEditor
8.          {
9.              get
10.             {
11.                 return Properties.Settings.Default.IsEditor;
12.             }
13.             set
14.             {
15.                 Properties.Settings.Default.IsEditor = value;
16.                 Properties.Settings.Default.Save();
17.             }
18.         }
19.     }
20. }
```

Dependency Injection i Microsoft Unity

Teraz kiedy mamy już wszystkie potrzebne interfejsy i klasy możemy zająć się głównym tematem. W module A w pliku ModuleA.cs modyfikujemy funkcję Initialize() oraz dodajemy nową.

```
1.  public void Initialize()
2.  {
3.      Container.RegisterInstance<IProductRepository>(new HardcodedProductRepository());
4.      Container.RegisterType<IAuthority, SettingsAuthority>();
5.
6.      Container.RegisterType<Object, EditorCatalog>(typeof(EditorCatalog).Name);
7.      Container.RegisterType<Object, Catalog>(typeof(Catalog).Name);
8.
9.      RegionManager.RegisterViewWithRegion("MainRegion", GetModuleView);
10.     RegionManager.RegisterViewWithRegion("MenuRegionA", typeof(ButtonViewA));
11. }
12.
13. public object GetModuleView()
14. {
15.   var auth = Container.Resolve<IAuthority>();
```

```
16.
17.   if (auth.IsEditor)
18.         return Container.Resolve<EditorCatalog>();
19.   else
20.         return Container.Resolve<Catalog>();
21. }
```

Modyfikujemy także plik `ButtonViewAViewModel.cs`

```
1.  using System.Text;
2.  using System.Threading.Tasks;
3.  using Microsoft.Practices.Prism.Commands;
4.  using Microsoft.Practices.Prism.Regions;
5.  using Microsoft.Practices.ServiceLocation;
6.  using Services;
7.  using ModuleA.Views;
8.  using ModuleA.ViewModels;
9.
10. namespace ModuleA
11. {
12.         class ButtonViewAViewModel
13.         {
14.                 private readonly IRegionManager regionManager;
15.                 private readonly IAuthority authority;
16.                 public DelegateCommand SwitchViewCommand { get; set; }
17.
18.                 public ButtonViewAViewModel()
19.                 {
20.                         regionManager = ServiceLocator.Current.GetInstance<IRegionManager>()
    ;
21.                         authority = ServiceLocator.Current.GetInstance<IAuthority>();
22.                         SwitchViewCommand = new DelegateCommand(SwitchView);
23.                 }
24.
25.                 private void SwitchView()
26.                 {
27.                         if (authority.IsEditor)
28.                                 regionManager.RequestNavigate("MainRegion", new Uri("EditorC
    atalog", UriKind.Relative));
29.                         else
30.                                 regionManager.RequestNavigate("MainRegion", new Uri("Catalog
    ", UriKind.Relative));
31.                 }
32.         }
```

```
33. }
```

W module B musimy dodać plik ButtonViewBViewModel.cs

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.Threading.Tasks;
6.  using Microsoft.Practices.Prism.Commands;
7.  using Microsoft.Practices.Prism.Regions;
8.  using Microsoft.Practices.ServiceLocation;
9.
10. namespace ModuleB
11. {
12.         public class ButtonViewBViewModel
13.         {
14.                 private readonly IRegionManager regionManager;
15.                 public DelegateCommand SwitchViewCommand { get; set; }
16.
17.                 public ButtonViewBViewModel()
18.                 {
19.                         regionManager = ServiceLocator.Current.GetInstance<IRegionManager>()
    ;
20.                         SwitchViewCommand = new DelegateCommand(SwitchView);
21.                 }
22.
23.                 private void SwitchView()
24.                 {
25.                         regionManager.RequestNavigate("MainRegion", new Uri("ModuleBView",
    UriKind.Relative));
26.                 }
27.         }
28. }
```