1. Tworzenie Shella
    a. W pierwszej kolejności tworzymy nowy projekt: WPF Application.
    Name: Shell
    SolutionName: PrismApp
    b. Dodajemy bibliotekę PRISM za pomocą NuGet Managera (dla .Net Framework 4.5 – Prism 5.0, dla starszych – Prism 4.0). Wyszukujemy w NuGetcie frazę „Prism" i dodajemy tę pozycję

2. Konfiguracja Shella
    a. Usuwamy MainWindow.xaml z projektu i dodajemy nowe okno (Window WPF) do projektu o nazwie ShellWindow.xaml
    b. Następnie dodajemy w XAMLu okna Shell definicję przestrzeni nazw:
    xmlns:prism=http://www.codeplex.com/prism
    c. Usuwamy Grida i dodajemy:
    <ContentControl prism:RegionManager.RegionName="MainRegion" />
    Ten kod oznacza że w przypadku kiedy zarejestrujemy widok w naszym module do regionu o nazwie MainRegion, to właśnie pokaże się on w tym miejscu.
    Shell może zawierać dowolną liczbę regionów, ustawionych w dowolny sposób. Jednakże nie do każdego rodzaju kontrolki można wstawić region.
    Prism podstawowo dostarcza 4 rodzaje tzw. region adapterów do których można przypisać region. Jednakże istnieje możliwość tworzenia własnych przez co możliwości tworzenia UI są nieograniczone.
    d. Pełen kod okna Shell:

```
<Window x:Class="Shell.ShellWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="ShellWindow" Height="300" Width="300"
        xmlns:prism="http://www.codeplex.com/prism">
        <ContentControl prism:RegionManager.RegionName="MainRegion" />
</Window>
```

3. Klasa inicjalizująca aplikację czyli Bootstrapper:
    a. Dodajemy klasę o nazwie Bootstrapper.cs, która będzie dziedziczyć po kontenerze (w tym wypadku UnityBootstrapper).
    By skorzystać z tego kontenera trzeba dodać pakiet NuGeta pod nazwą: Prism.UnityExtensions!
    b. Następnie musimy przeciążyć metody odpowiedzialne za inicjalizacje i stworzenie naszego okna Shell-a

```
protected override DependencyObject CreateShell()
{
        return new ShellWindow();
}
protected override void InitializeShell()
{
        base.InitializeShell();
        App.Current.MainWindow = (Window)this.Shell;
        App.Current.MainWindow.Show();
}
```

    c. Ostatnia metoda jaką musimy dodać to przeciążona metoda konfiguracji katalogu modułów, jako że na tym etapie nie dodajemy żadnego modułu to póki co zostanie ona pusta.

```
protected override void ConfigureModuleCatalog()
{
        base.ConfigureModuleCatalog();

}
```

d. Ostatnią rzeczą jaką musimy zrobić, to zapewnić aby podczas uruchamiania naszej aplikacji pierwsza uruchamiała się klasa Bootstrapera
   - Otwieramy plik App.xaml.cs i przeciążamy metodę OnStartup.
     protected override void OnStartup(StartupEventArgs e)
     {
     base.OnStartup(e);
     Bootstrapper bootstrapper = new Bootstrapper();
     bootstrapper.Run();
     }
     Otwieramy plik App.xaml i warto pokazać że uruchamiają się dwa okna zmieniając na **StartupUri="ShellWindow.xaml"**
   - Usuwamy znacznik **StartupUri="ShellWindow.xaml"**
     Po tych zabiega powinniśmy mieć gotowy bootstraper.

4 Dodanie pierwszego modułu
   a. Dodajemy nowy projekt do solucji, nazwijmy go ModuleA, jak typ projektu wybieramy WPF User Control Libary. Po jego utworzeniu usuwamy domyślnie stworzoną kontrolkę UserControl1.xaml.
   b. Dodajemy referencje Prisma oraz Prism.UnityExtensions poprzez konfiguracje pakietu Nugeta do nowo utworzonego projektu modułu.
   c. W projekcie modułu dodajemy nową klasę ModuleA.cs, robimy ja publiczną oraz implementujemy interfejs IModule. Jest to główna klasa naszego modułu, w której to będziemy rejestrować nasze widoki, view modele, usługi itd.
   d. Dodajemy zmienną RegionManager typu IRegionManager, która posłuży nam do rejestrowania widoków do odpowiednich regionów w shellu.
   e. Tworzymy konstruktor który przyjmuje wstrzyknięty IRegionManager, następnie w konstruktorze przypisujemy wcześniej wstrzyknięty regionManager do naszej zmiennej.

```
public class ModuleA : IModule
{
        protected IRegionManager RegionManager { get; private set; }
        public ModuleA(IRegionManager regionManager)
        {
                RegionManager = regionManager;
        }
        public void Initialize()
        {
                throw new NotImplementedException();
        }
}
```

   Póki co metodę Initialize() zostawiamy do czasu aż nie stworzymy widoku.

5 Dodajemy widok modułu.
   a. Do projektu modułu dodajemy User Control (WPF) i nazywamy ją np.: ModuleAView
   b. Edytujemy XAMLa dodanej kontrolki dodając np.:
      <Label Content="Widok z Modułu A" FontSize="15" Background="Aquamarine" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>
   c. Wracamy do metody Initialize w klasie ModuleA i za pomocą wstrzykniętego Region Managera rejestrujemy widok.

```
public void Initialize()
{
        RegionManager.RegisterViewWithRegion("MainRegion", typeof(ModuleAView));
}
```

   Tak zarejestrowany widok pokaże się automatycznie zaraz po inicjalizacji modułu. Jednakże póki co jeszcze nic nie zobaczymy ponieważ trzeba zarejestrować nasz moduł w klasie bootstrappera.

d. Wracamy do klasy bootstrappera i zmieniamy metodę ConfigureModuleCatalog.

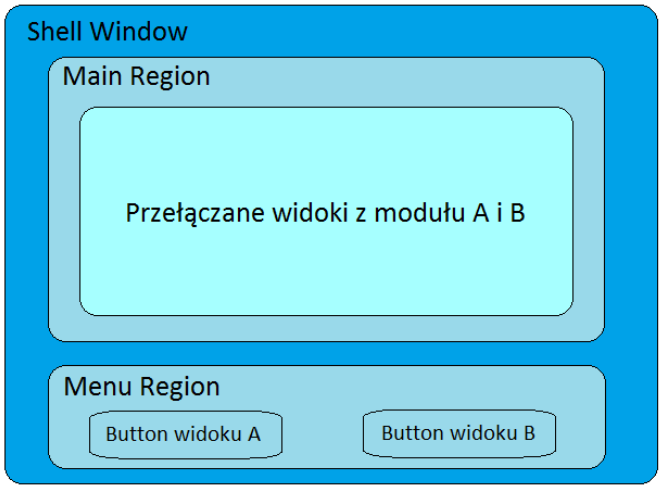
```
protected override void ConfigureModuleCatalog()
{
        base.ConfigureModuleCatalog();
        ModuleCatalog moduleCatalog = (ModuleCatalog)this.ModuleCatalog;
        moduleCatalog.AddModule(typeof(ModuleA.ModuleA));
}
```

Trzeba też nadmienić że istnieją jeszcze inne sposoby rejestrowania modułów, pokazana tutaj metoda nie jest najbardziej "elegancka" gdyż wymaga dodania refencji modułu do shella. Z innych metod dodawania to min: rejestrowanie modułów z fizycznego folderu na dysku, czy też z plików konfiguracyjnych.

Aby wszystko działało musimy dodać referencje w naszym Shellu i podpiąć nasz ModuleA oraz dodać przestrzeń nazw obsługującą moduły z biblioteki Prism.

6 Dodawanie drugiego modułu i przełączania pomiędzy nimi.

Nasza aplikacja powinna posiadać menu i będzie nim dodatkowy widok w naszym module lecz wyświetlany w innym regionie. By przełączać się powinniśmy posiadać inne widoki lub widoki w ogóle innym module, tak też zrobimy. Tak będzie wyglądać nasza aplikacja:



a. Na początek trzeba edytować Shell i dodać region w którym będziemy wyświetlać widoki przycisków przełączających widok w naszym głównym regionie o nazwie Main Region.

```
<Grid>
        <Grid.RowDefinitions>
                <RowDefinition Height="5*"/>
                <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
                <ColumnDefinition Width="1*"/>
                <ColumnDefinition Width="1*"/>
        </Grid.ColumnDefinitions>
        <ContentControl Grid.Row="0" Grid.ColumnSpan="2"
        prism:RegionManager.RegionName="MainRegion" />
        <ContentControl Grid.Row="1" Grid.Column="0"
        prism:RegionManager.RegionName="MenuRegionA" />
        <ContentControl Grid.Row="1" Grid.Column="1"
        prism:RegionManager.RegionName="MenuRegionB" />
</Grid>
```

b. Następnie do projektu modułu A dodajemy widok (Add User Control – WPF), który będzie elementem naszego menu o nazwie ButtonViewA, będzie to zwykły buton pod który później

podepniemy komendę. Po jego dodaniu zarejestrujmy go w klasie modułu, tak aby był widoczny od razu po jego załadowaniu. Tym razem ten widok rejestrujemy do regionu o nazwie MenuRegionA, który skonfigurowaliśmy przed chwilą w Shellu. Zmieniamy również sposób w jaki będziemy pokazywać nasz widok w regionie MainRegion. Zamiast rejestrować go od razu do region managera zarejestrujemy go do kontenera, którego instancje wcześniej musimy pobrać chociażby np. w konstruktorze.

```
public class ModuleA : IModule
{
        protected IRegionManager RegionManager { get; private set; }
        protected IUnityContainer Container { get; private set; }

        public ModuleA(IRegionManager regionManager, IUnityContainer container)
        {
                Container = container;
                RegionManager = regionManager;
        }

        public void Initialize()
        {
                Container.RegisterType<Object, ModuleAView>(typeof(ModuleAView).Name);
                RegionManager.RegisterViewWithRegion("MenuRegionA", typeof(ButtonViewA));
        }
}
```

c. Dodajemy teraz projekt drugiego modułu analogicznie jak w przypadku modułu A(User Control Library). Nazwijmy go ModuleB i nie zapomnijmy dodać go do katalogu modułów w klasie bootstrappera.
moduleCatalog.AddModule(typeof(ModuleB.ModuleB));
Uprzednio dodając referencje naszego modułu do projektu Shell oraz do samego modułu referencje Prisma i Prism.UnityExtensions.
Dodajmy do niego również klasę i widoki podobnie jak mamy je zorganizowane w pierwszym module i tak samo zarejestrujemy w metodzie Initialize.
Kod widoku ModuleBView:
<Label Content="Widok z Modułu B" FontSize="15" Background="BurlyWood" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>
Dodany został inny kolor tak aby można było odróżniać widoki.

d. Tak skonfigurowaną aplikację mamy prawie gotową. Pozostał jeszcze ważny element czyli View Modele do widoków gdzie będziemy bindować nasze komendy przełączające widoki.
Poniższe czynności wykonujemy dla każdego z modułów.

- Dodajemy klasę view modelu i wykonujemy w niej następujące czynności
    1. Pobieramy instancję region menadżera z pomocą ServiceLocatora
    2. Tworzymy komendę typu DelegateCommand i inicjalizujemy ją w konstruktorze view modelu.
    3. Do naszej komendy dodajemy metodę SwitchView, w ciele tej metody wykonujemy metodę region managera RequestNavigate w której okeślamy do jakiego regionu wstawiamy widok oraz jaki widok określany po jego nazwie użytej podczas rejestrowania widoku kontenera.

```
public class ButtonViewBViewModel
{
        private readonly IRegionManager regionManager;
        public DelegateCommand SwitchViewCommand { get; set; }
```

```
        public ButtonViewBViewModel()
        {
            regionManager =
            ServiceLocator.Current.GetInstance<IRegionManager>();
            SwitchViewCommand = new DelegateCommand(SwitchView);
        }

        private void SwitchView()
        {
        regionManager.RequestNavigate("MainRegion", new Uri("ModuleBView",
        UriKind.Relative));
        }
    }
```

Nie trzeba dziedziczyć po INotifyPropertyChanged ponieważ Prism robi to za nas.
Łączymy nasz view model z Data Contextem naszego widoku:

```
public partial class ButtonViewB : UserControl
{
    public ButtonViewB()
    {
        InitializeComponent();
        this.DataContext = new ButtonViewBViewModel();
    }
}
```

- i ostatecznie bindujemy naszą komendę w XAMLu:
  <Button Content="Przełącz na widok B" Command="{Binding SwitchViewCommand}" />

Te same czynności powtarzamy dla modułu A. Nasza aplikacja skończona. ☺
Notka: Jeśli chcemy by przy odpalaniu aplikacji pojawiał się domyślnie widok z modułu A to w metodzie Initialize modułu A dodajemy dodatkową linię:
RegionManager.RegisterViewWithRegion("MainRegion", typeof(ModuleAView));
Można tę linijkę dodać również w Module B i zamienić kolejność inicjacji modułów w Bootstraperze na:

```
protected override void ConfigureModuleCatalog()
{
    //[…]
    moduleCatalog.AddModule(typeof(ModuleB.ModuleB));
    moduleCatalog.AddModule(typeof(ModuleA.ModuleA));
}
```

Widać wtedy, że ta kolejność decyduje o tym jaki widok zostanie wczytany.
Koniec ☺