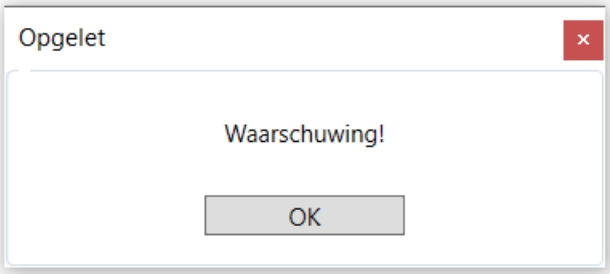
HOW TO –
F. WPF MVVM Dialogen gebruiken

(Voorbeeldapplicatie van bieren, Brouwers en soorten bieren)

https://github.com/CSharpSyntraWest/HOWTO-WPF-MVVM

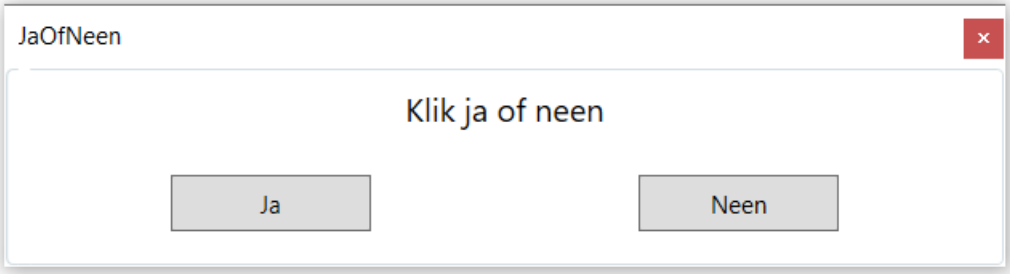
1. **In dit hoofdstuk wordt er getoond hoe je eenvoudige dialoogvensters kan tonen in popup. Hiervoor gebruiken we code die normaal gezien door een MVVM framework wordt voorzien. Deze moet je dus niet zelf kunnen schrijven.**
   De volgende Dialoogviews worden getoond:
   AlertDialog, YesNoDialog en InputDialog.

   AlertDialog :
   Deze toont een waarschuwing en ok button:

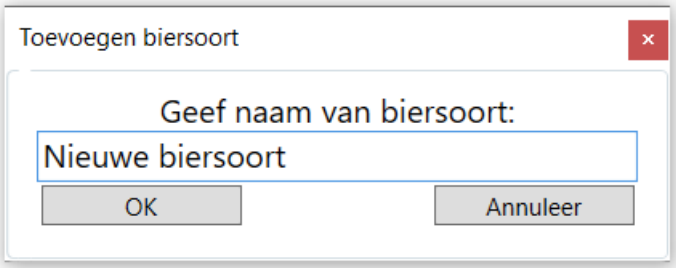


   YesNoDialog:
   Deze toont een vraag en een Ja of Neen button. Bv Bent u zeker dat u dit bier wil verwijderen ?

   Bv:



InputDialog kan er een tekst worden ingegeven (bv om een nieuwe biersoort toe te voegen) :



1) **Voeg onder Services een abstract class `DialogViewModelBase<T>` toe die afgeleid is van ObservableObject. Alle Viewmodel classes voor Dialoogventers zullen afgeleid worden van deze `DialogViewModelBase<T>` class.**

```
public abstract class DialogViewModelBase<T>:ObservableObject
{
    public string Title { get; set; }
    public string Message { get; set; }
    public T DialogResult { get; set; }

    public DialogViewModelBase() : this(string.Empty, string.Empty) { }
    public DialogViewModelBase(string title) : this(title, string.Empty) {     }
    public DialogViewModelBase(string title, string message)
    {
        Title = title;
        Message = message;
    }
    public void CloseDialogWithResult(IDialogWindow dialog, T result)
    {
        DialogResult = result;
        if (dialog != null)
            dialog.DialogResult = true;
    }
}
```

2) **Voeg een public enum DialogResults toe onder services in een apart bestand DialogResults.cs. Dit opsommingstype zal worden gebruikt door de Dialog vensters om een antwoord terug te geven vanuit het dialoogvenster**

```
public enum DialogResults
{
    Undefined,
    Yes,
    No,
    Cancel
}
```

3) **Voeg de volgende public interface toe onder de folder Services :**

```
public interface IDialogService
{
    T OpenDialog<T>(DialogViewModelBase<T> viewModel);
}
```

4) **Voeg onder de folder nog een interface `IDialogWindow` toe, deze definieert 2 auto-properties DataContext, om een DataContext te kunnen instellen en lezen, DialogResult, die true of false teruggeeft en 1 ShowDialog() methode:**

```
public interface IDialogWindow
{
    bool? DialogResult { get; set; }
    object DataContext { get; set; }
    bool? ShowDialog();
}
```

5) **Voeg de Window `DialogWindow.xaml` toe onder de folder Services. Deze bevat een ContentControl die zal worden ingevuld met een dialogview usercontrol en zal worden gebonden met de bijpassend Dialogviewmodel**

```
<Window x:Class="F_DialogWindowWPFMVVM.Services.DialogWindow"
        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:local="clr-namespace:F_DialogWindowWPFMVVM.Views"
        mc:Ignorable="d"
        SizeToContent="WidthAndHeight"
        WindowStartupLocation="CenterScreen"
        WindowStyle="ToolWindow"
        ResizeMode="NoResize"
        Title="{Binding Title}" Height="450" Width="800" >
    <ContentControl x:Name="ContentPresenter" Content="{Binding}"/>
</Window>
```

6) **Open de Code behind van DialogWindow: DialogWindow.xaml.cs en voeg de interface `IDialogWindow` ,toe zodat DialogWindow deze interface implementeert**

```
public partial class DialogWindow : Window,IDialogWindow
{
    public DialogWindow()
    {
        InitializeComponent();
    }
}
```

7) **Voeg de class Service toe onder de folder Services die de interface IDialogService implementeert. Deze maakt ook gebruik van IDialogWindow interface om een nieuw object van de klasse DialogWindow in bij te houden**

```
public class DialogService : IDialogService
{
    public T OpenDialog<T>(DialogViewModelBase<T> viewModel)
    {
        IDialogWindow window = new DialogWindow();
        window.DataContext = viewModel;
        window.ShowDialog();
        return viewModel.DialogResult;
    }
}
```

8) **Voeg onder de folder ViewModels de volgende classen toe voor een AlertDialog, een YesNoDialog en InputDialog:**

AlertDialogViewModel.cs

```
public class AlertDialogViewModel:DialogViewModelBase<DialogResults>
{
    public ICommand OKCommand { get; private set; }
    public AlertDialogViewModel(string title, string message):base(title,message)
    {
```

```
                OKCommand = new RelayCommand<IDialogWindow>(OK);
            }
            private void OK(IDialogWindow window)
            {
                CloseDialogWithResult(window, DialogResult);
            }
        }
```

YesNoDialogViewModel.cs

```
    public class YesNoDialogViewModel : DialogViewModelBase<DialogResults>
    {
        public ICommand YesCommand { get; private set; }
        public ICommand NoCommand { get; private set; }

        public YesNoDialogViewModel(string title, string message) : base(title, message)
        {
            YesCommand = new RelayCommand<IDialogWindow>(Yes);
            NoCommand = new RelayCommand<IDialogWindow>(No);
        }

        private void Yes(IDialogWindow window)
        {
            CloseDialogWithResult(window, DialogResults.Yes);
        }

        private void No(IDialogWindow window)
        {
            CloseDialogWithResult(window, DialogResults.No);
        }
    }
```

InputDialogViewModel.cs

```
    public class InputDialogViewModel : DialogViewModelBase<object>
    {
        public ICommand OKCommand { get; private set; }
        public ICommand CancelCommand { get; private set; }
        public string _answer;


        public InputDialogViewModel(string title, string message) : base(title, message)
        {
            OKCommand = new RelayCommand<IDialogWindow>(Ok);
            CancelCommand = new RelayCommand<IDialogWindow>(Cancel);
        }
        public string Answer
        {
            get { return _answer; }
            set {
                OnPropertyChanged(ref _answer, value);
            }
        }
        private void Ok(IDialogWindow window)
        {
            CloseDialogWithResult(window, Answer);
        }

        private void Cancel(IDialogWindow window)
        {
            CloseDialogWithResult(window, DialogResults.Cancel);
        }
    }
```

9) Voeg onder de folder Views de usercontrols AlertDialogView, een YesNoDialogView en InputDialogView toe. Aan de dialogViews voegen we een CommandParameter toe, deze verwijst naar de parentwindow object waarin de button staat en wordt gebruikt om de window af te sluiten

AlertDialogView.xaml

```
<UserControl x:Class="F_DialogWindowWPFMVVM.Views.AlertDialogView"
             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:local="clr-namespace:F_DialogWindowWPFMVVM.Views"
             mc:Ignorable="d"
             Height="100" Width="300">
    <GroupBox>
    <Grid Margin="10">
        <Grid.RowDefinitions>
                <RowDefinition />
                <RowDefinition Height="10"/>
                <RowDefinition Height="auto" />
        </Grid.RowDefinitions>

        <TextBlock Text="{Binding Message}" Grid.Row="0"
                   HorizontalAlignment="Center" VerticalAlignment="Center"
                   TextWrapping="Wrap" />
        <Button Content="OK" Grid.Row="2" Width="100"
                Command="{Binding OKCommand}"
                CommandParameter="{Binding RelativeSource={RelativeSource Mode=FindAncestor,
            AncestorType=Window}}" />
    </Grid>
    </GroupBox>
</UserControl>
```

YesNoDialogView.xaml

```
<UserControl x:Class="F_DialogWindowWPFMVVM.Views.YesNoDialogView"
             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:local="clr-namespace:F_DialogWindowWPFMVVM.Views"
             mc:Ignorable="d"
             Height="100" Width="500">
    <GroupBox>
        <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition Height="10" />
            <RowDefinition />
        </Grid.RowDefinitions>

        <TextBlock Text="{Binding Message}" Grid.Row="0" HorizontalAlignment="Center" FontSize="16" />

        <Grid Grid.Row="2">
            <Grid.ColumnDefinitions>
                <ColumnDefinition />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>

            <Button Content="Ja" Command="{Binding YesCommand}" Grid.Column="0" Margin="2"
                CommandParameter="{Binding RelativeSource={RelativeSource Mode=FindAncestor,
                AncestorType=Window}}" Width="100"/>
            <Button Content="Neen" Command="{Binding NoCommand}" Grid.Column="1" Margin="2"
                CommandParameter="{Binding RelativeSource={RelativeSource Mode=FindAncestor,
                AncestorType=Window}}" Width="100" />
        </Grid>
    </Grid>
    </GroupBox>

</UserControl>
```

InputDialogView.xaml

```
<UserControl x:Class="F_DialogWindowWPFMVVM.Views.InputDialogView"
             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:local="clr-namespace:F_DialogWindowWPFMVVM.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <GroupBox>
    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition Height="25" />
            <RowDefinition />
        </Grid.RowDefinitions>

        <TextBlock Grid.Row="0" Text="{Binding Message}"  HorizontalAlignment="Center" FontSize="16" />
        <TextBox Grid.Row="1" Text="{Binding Answer}"  MinWidth="300" HorizontalAlignment="Center" FontSize="16" />
        <Grid Grid.Row="2">
            <Grid.ColumnDefinitions>
                <ColumnDefinition />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>

            <Button Width="100" HorizontalAlignment="Left" Content="OK" Command="{Binding OKCommand}" Grid.Column="0" Margin="2"
                CommandParameter="{Binding RelativeSource={RelativeSource Mode=FindAncestor,
                AncestorType=Window}}" />
            <Button Width ="100" HorizontalAlignment="Right" Content="Annuleer" Command="{Binding CancelCommand}" Grid.Column="1"
Margin="2"
                CommandParameter="{Binding RelativeSource={RelativeSource Mode=FindAncestor,
                AncestorType=Window}}" />
        </Grid>
        </Grid>
    </GroupBox>
</UserControl>
```

10) Open MainWindow.xaml en voeg 2 buttons toe (in linkerkolom, boven de eerste expander). We binden de Command van de buttons aan AlertCommand en YesNoCommand

```
            <StackPanel>
                <Button Margin="10" Content="Test alert" Command="{Binding AlertCommand}"/>
                <Button Margin="10" Content="Test Ja/Neen dialog" Command="{Binding YesNoCommand}"/>
                <Expander Header="Bieren">
                    <views:BierenView DataContext="{Binding BierenVM}" Margin="10"/>
                </Expander>
```

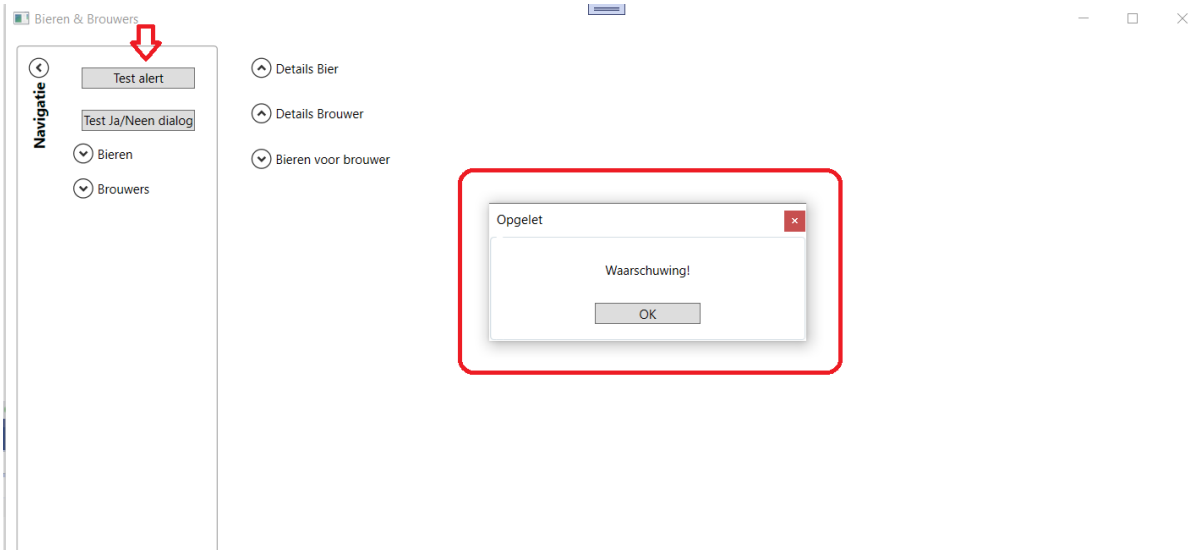
11) Open MainWindowViewModel.cs en voeg AlertCommand en YesNoCommand als public ICommand properties en initialiseer deze in de constructor. Definieer eveneens een private field _dialogService en initialiseer deze eveneens in de constructor

```
public class MainViewModel : ObservableObject
{
    private IDialogService _dialogService;
    private IDataService _dataService;
    private BierenViewModel _bierenVM;
    private BrouwersViewModel _brouwersVM;
    public MainViewModel()
    {
        _dialogService = new DialogService();
        _dataService = new MockDataService();
        BierenVM = new BierenViewModel(_dataService);//,_dialogService);
        BrouwersVM = new BrouwersViewModel(_dataService);
        AlertCommand = new RelayCommand(ShowAlert);
        YesNoCommand = new RelayCommand(ShowYesNoDialog);

    }

    public ICommand AlertCommand { get; private set; }
    public ICommand YesNoCommand { get; private set; }
```

12) Voeg de methoden `ShowAlert` en `ShowYesNoDialog` toe aan MainViewModel. Deze zullen de Dialog AlertDialogView en YesNoDialogView tonen in een popup via de methode `OpenDialog();`

```
private void ShowYesNoDialog()
{
    YesNoDialogViewModel dialog = new YesNoDialogViewModel("JaOfNeen", "Klik ja of neen");
    DialogResults result = _dialogService.OpenDialog(dialog);
    Debug.WriteLine(result);
}

private void ShowAlert()
{
    AlertDialogViewModel dialog = new AlertDialogViewModel("Opgelet", "Waarschuwing!");
    DialogResults result = _dialogService.OpenDialog(dialog);
    Debug.WriteLine(result);
}
```

13) Test de app en klik op de button « Test alert »



14) en klik op de button « Test Ja/Neen dialog »