Closing a Window in an MVVM WPF application

Implemented an `CloseWindow` Method which takes a Windows as parameter and close it. The window is passed to the ViewModel via `CommandParameter`. Note that you need to define an `x:Name`for the window which should be close. In my XAML Window i call this method via `Command` and pass the window itself as a parameter to the ViewModel using `CommandParameter`.

```
Command="{Binding CloseWindowCommand, Mode=OneWay}"
CommandParameter="{Binding ElementName=TestWindow}"
```

It is a clean and easy solution which is in compliance with the MVVM programming paradigm.

ViewModel

```
public RelayCommand<Window> CloseWindowCommand { get; private set; }

public MainViewModel()
{
   this.CloseWindowCommand = new RelayCommand<Window>(this.CloseWindow);
}

private void CloseWindow(Window window)
{
   if (window != null)
   {
     window.Close();
   }
}
```

XAML

```
Window x:Class="ClientLibTestTool.ErrorView"
      x:Name="TestWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:localization="clr-namespace:ClientLibTestTool.ViewLanguages"
      DataContext="{Binding Main, Source={StaticResource Locator}}"
      Title="{x:Static localization:localization.HeaderErrorView}" Height="600" Width="800"
ResizeMode="NoResize" WindowStartupLocation="CenterScreen">
   <Grid>
      <Button Content="{x:Static localization:localization.ButtonClose}"
            Height="30"
            Width="100"
            Margin="0,0,10,10"
            IsCancel="True"
            VerticalAlignment="Bottom"
            HorizontalAlignment="Right"
            Command="{Binding CloseWindowCommand, Mode=OneWay}"
```

CommandParameter="{Binding ElementName=TestWindow}"/>
```
    </Grid>
</Window>
```

Passing the Window object to the view model breaks the MVVM pattern IMHO, because it forces your vm to know what it's being viewed in.
You can fix this by introducing an interface containing a close method.

*Interface:*public interface IClosable
```
{    void Close();
}
```

Your refactored ViewModel will look like this:

ViewModel

```
public RelayCommand<IClosable> CloseWindowCommand { get; private set; }

public MainViewModel()
{

    this.CloseWindowCommand = new RelayCommand<IClosable>(this.CloseWindow);

}

private void CloseWindow(IClosable window)

{    if (window != null)
    {

        window.Close();

    }

}
public RelayCommand<IClosable> CloseWindowCommand { get; private set; }
    }
}
```

You need reference/implement the `IClosable` interface your view as well *View (Code behind)*

```
public partial class MainWindow : Window, IClosable
{    public MainWindow()
    {

        InitializeComponent();

    }

}
```