

# Hooking Up Views and ViewModels in MVVM



Brian Noyes

@briannoyes | [www.solliance.net](http://www.solliance.net)

# Overview



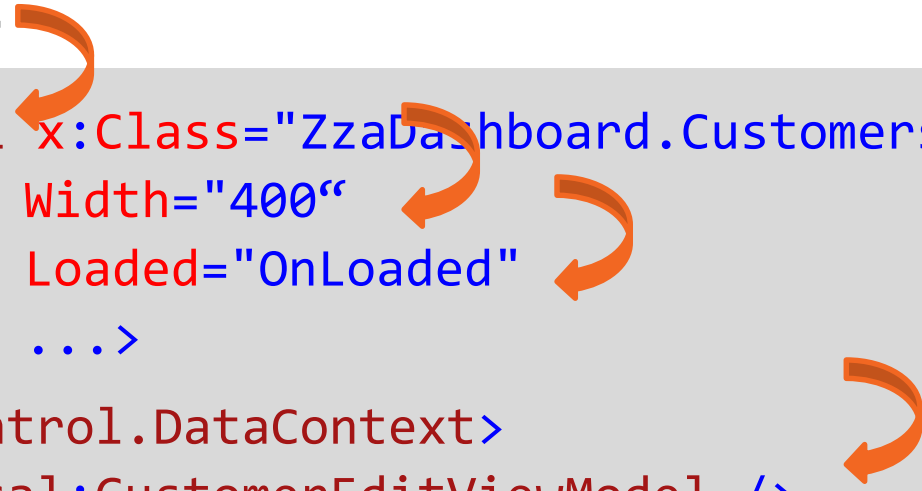
View-First Construction Patterns

Data Binding

ViewModel-First with DataTemplates

# View-First - XAML

InitializeComponent()

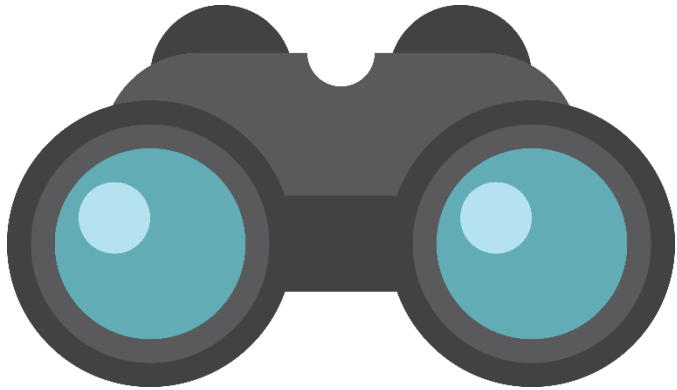


```
<UserControl x:Class="ZzaDashboard.Customers.CustomerEditView"  
    Width="400"  
    Loaded="OnLoaded"  
    ...>  
    <UserControl.DataContext>  
        <local:CustomerEditViewModel />  
    </UserControl.DataContext>  
    ...  
</UserControl>
```

# View-First Code-Behind

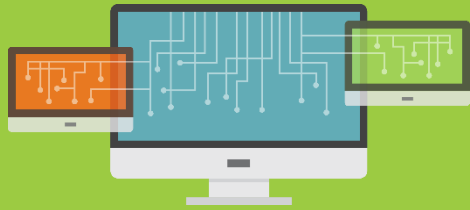
```
public partial class CustomerListView : UserControl
{
    public CustomerListView()
    {
        this.DataContext = new CustomerListViewModel();
        InitializeComponent();
    }
}
```

# View-First - ViewModelLocator



ViewModelLocator is a meta-pattern for automatically locating and hooking up the right ViewModel

# ViewModelLocator Process



What View is being constructed?



What ViewModel should I construct?

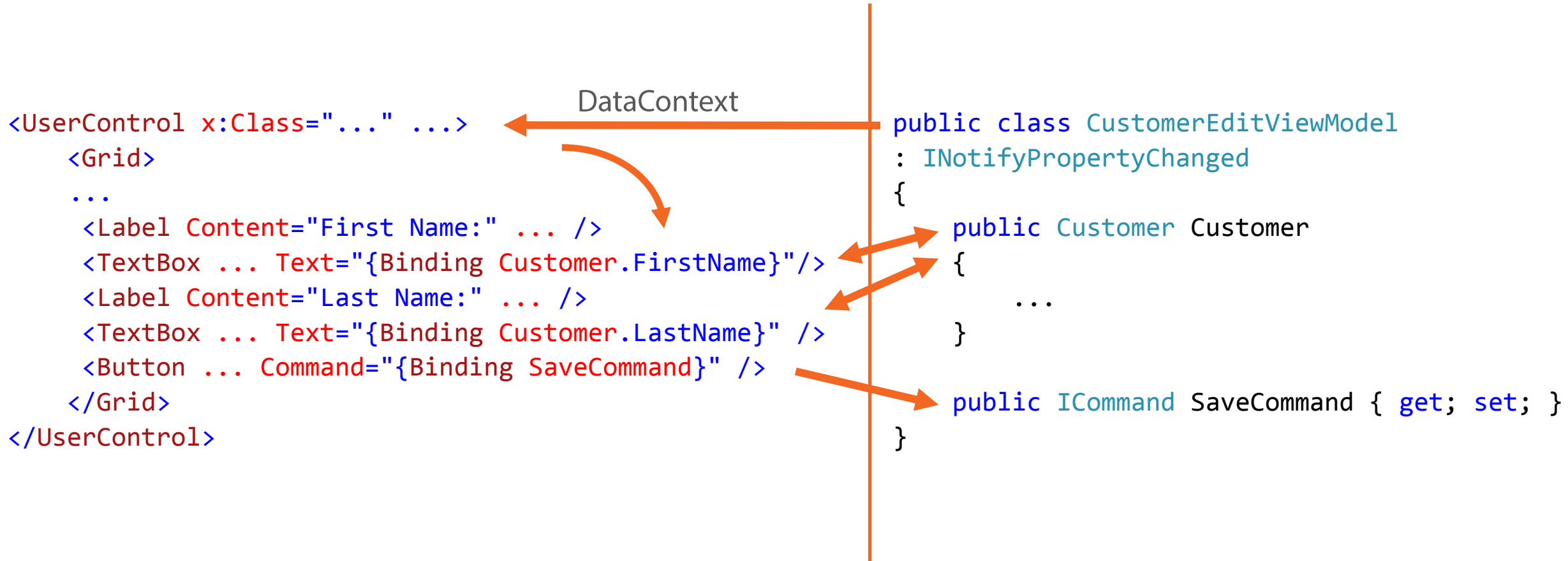


Construct ViewModel



Set DataContext

# Data Binding

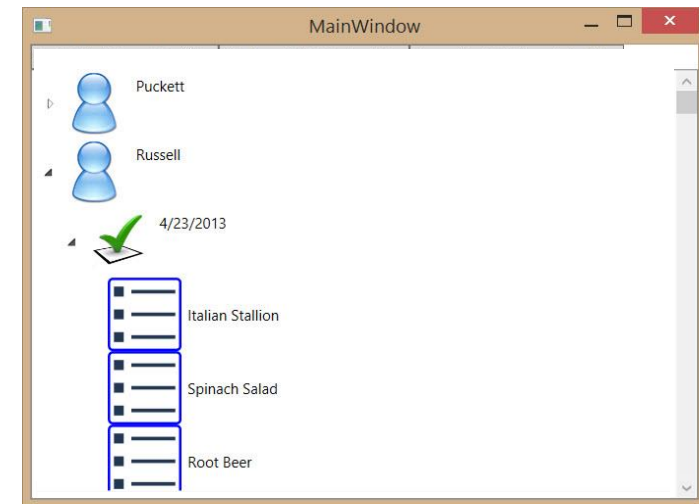


# ViewModel-First with DataTemplates

CustomersOrderTreeViewModel

```
<ContentControl Content="{Binding CurrentViewModel}" />
```

```
<DataTemplate DataType="{x:Type local:CustomersOrderTreeViewModel}">  
    <local:CustomersOrderTreeView />  
</DataTemplate>
```



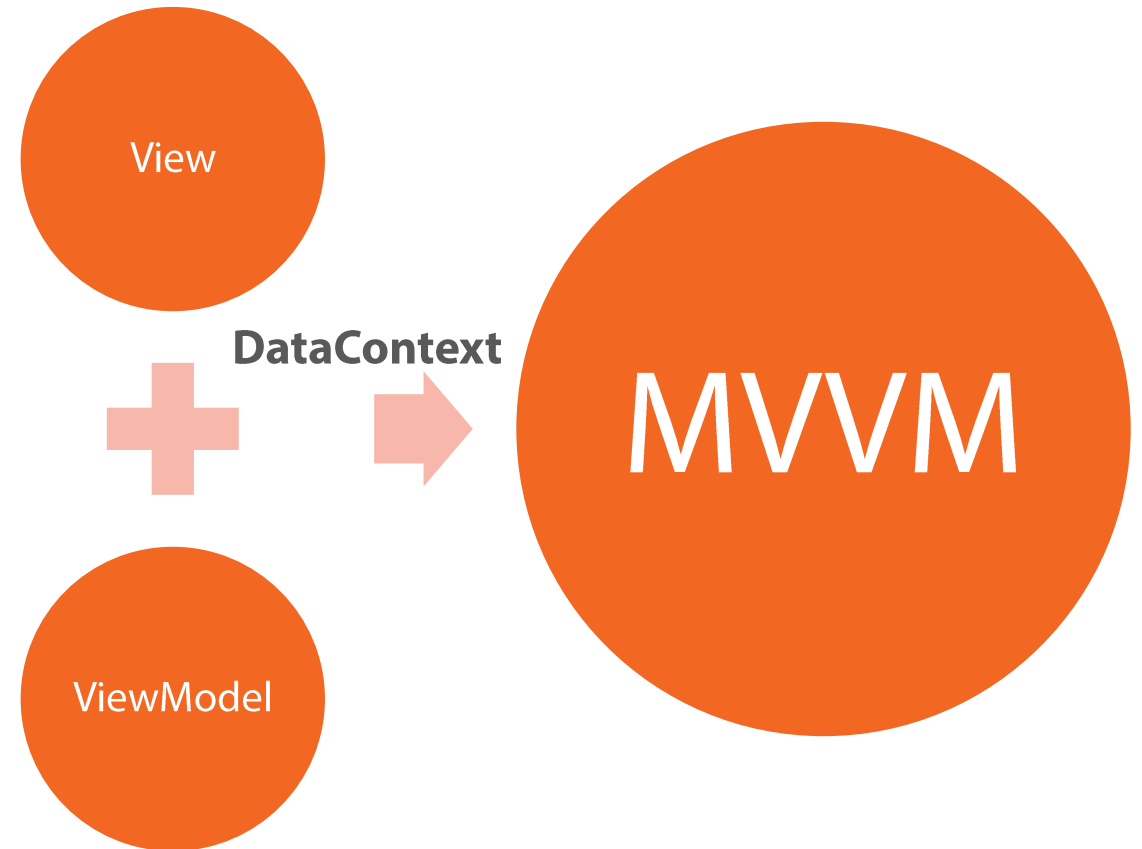


# No One's on First

May have other code in control of constructing both View and ViewModel

Order does not matter

Set DataContext after constructing both



# Summary



View-First MVVM hookup can be achieved several ways

Data binding forms the glue that flows

DataTemplates allow dynamic selection and hookup of ViewModels for Views