Applied MVVM - Hierarchies and Navigation

Brian Noyes
CTO AND CO-FOUNDER, SOLLIANCE INC

@briannoyes www.briannoyes.com

Introduction

Naming and locating components

Hierarchical MVVM & navigation

MVVM app building

There are no correct names, only good or bad ones in the eye of the beholder.

What matters is having a pattern and using it consistently.

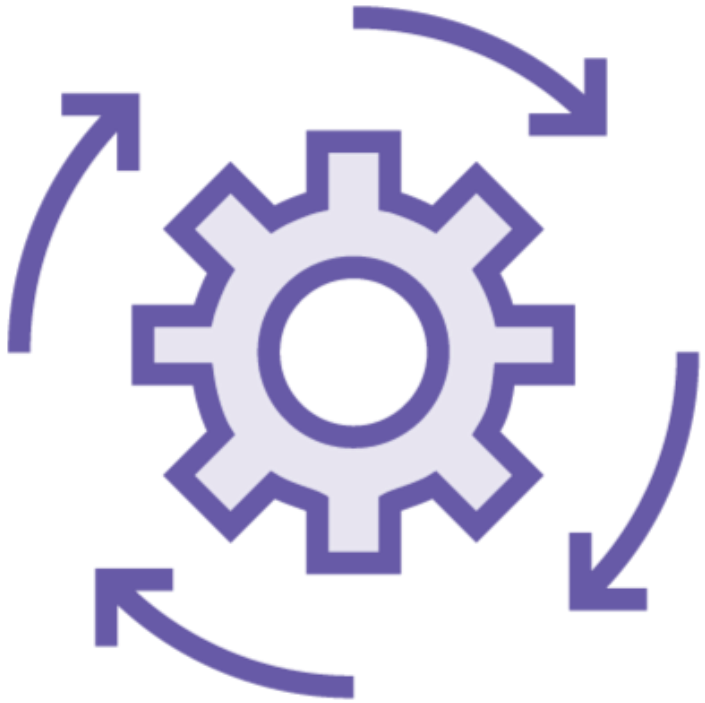
View Naming Guidance

Typically named “<Name>View”

May have other name suffixes (i.e. Window, Page, Dialog)

Recommendation: Don’t include “View” in the type name unless it has a corresponding ViewModel

ViewModel Naming Guidance



If the View name ends in “View”, append “Model”

- CustomerEditView
- CustomerEditViewModel

If the View name does not end in “View”, append “ViewModel”

- MainWindow
- MainWindowViewModel

The rest of the ViewModel name should match the View name

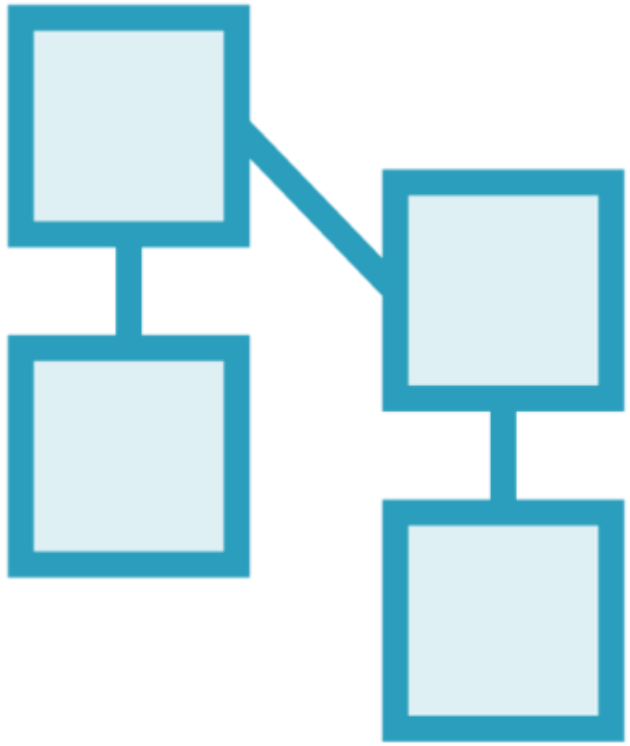
Naming Models

Model objects are mostly data objects or Plain Old CLR Objects (POCOs) – aka "Entities"

- Does not imply Entity Framework

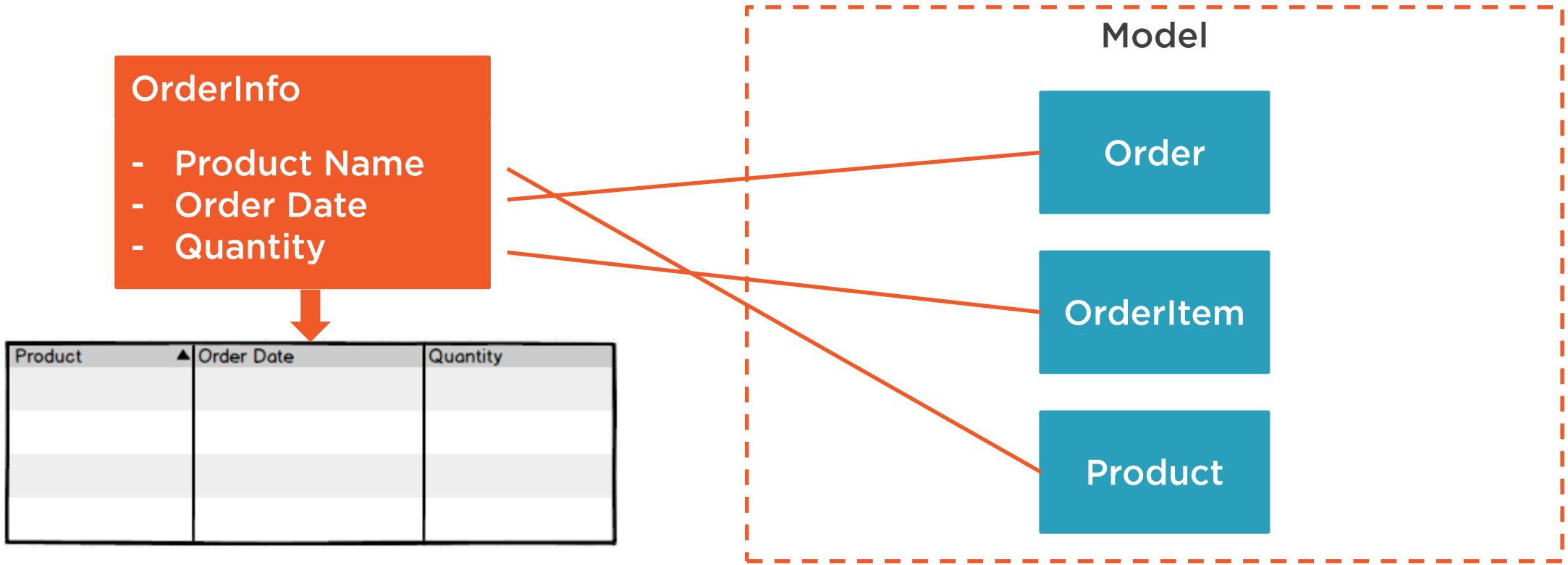
Name them for the data or state that they contain

- Customer, Order, Product, Patient, Prescription, etc.



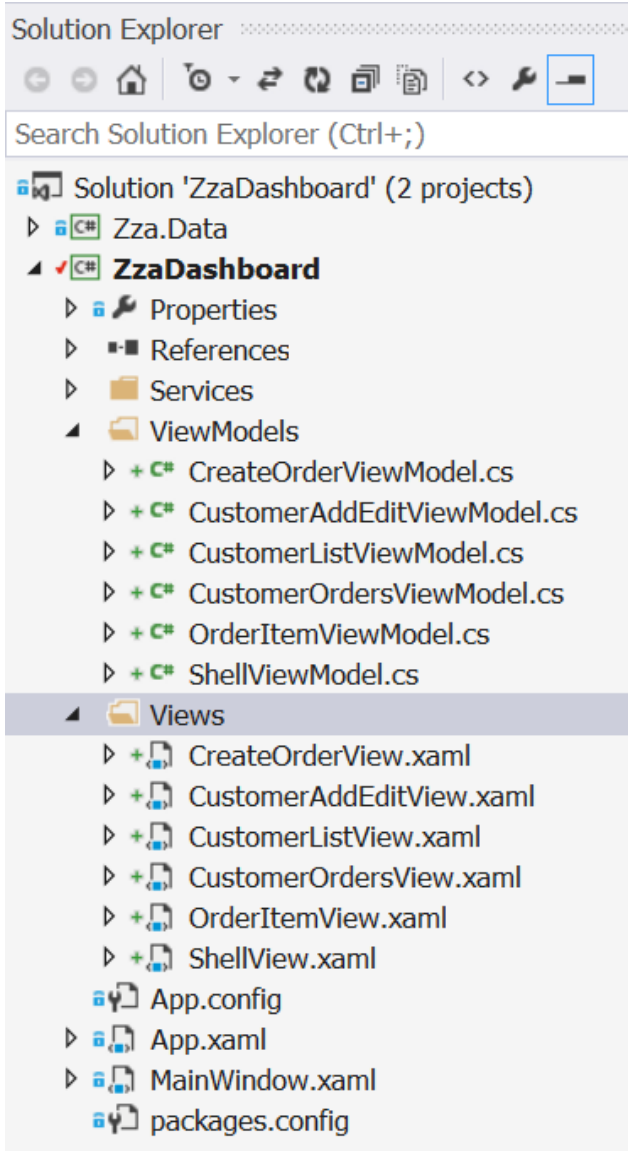
Wrapped Model Naming Guidance

- Don’t name data objects that don’t have a corresponding View a “ViewModel”
- But put them somewhere different from your Model types

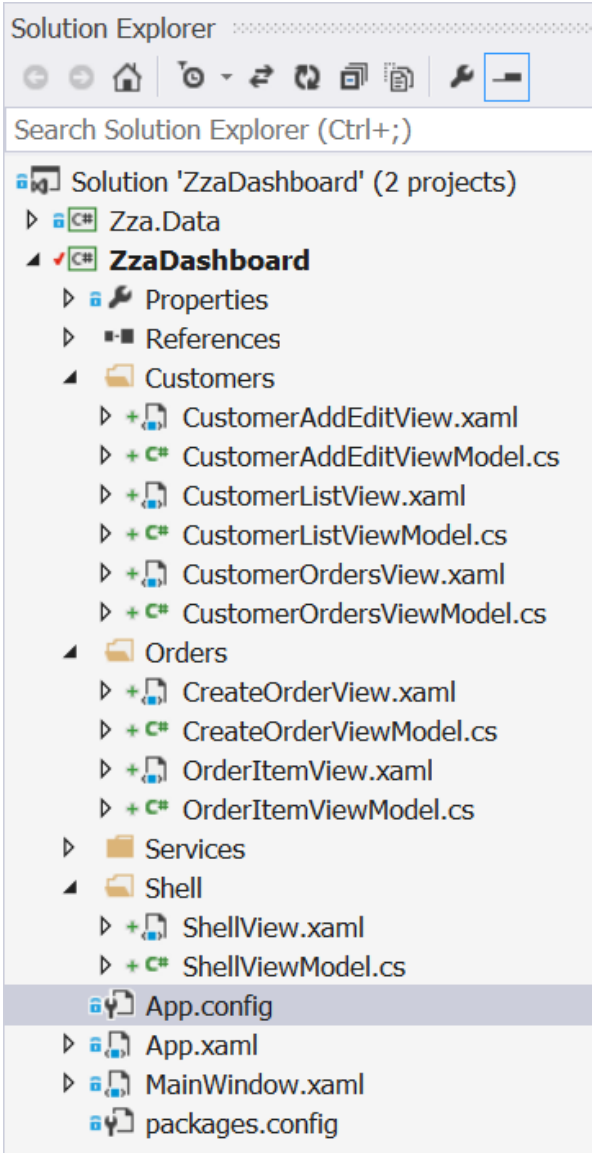


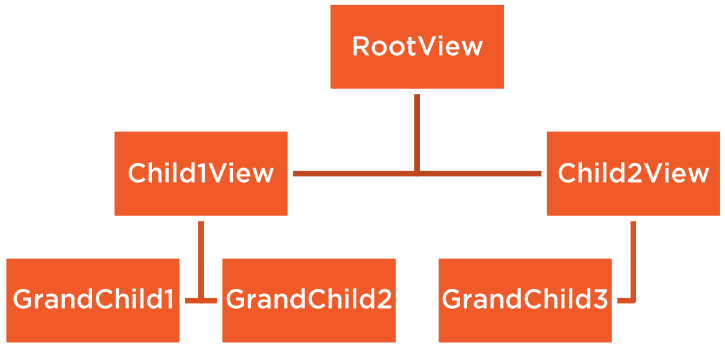
Locating Components

By Type:



By Feature:




RootView
Child1View
Child2View
GrandChild1
GrandChild2
GrandChild3

Hierarchical MVVM and Navigation

Views are commonly nested

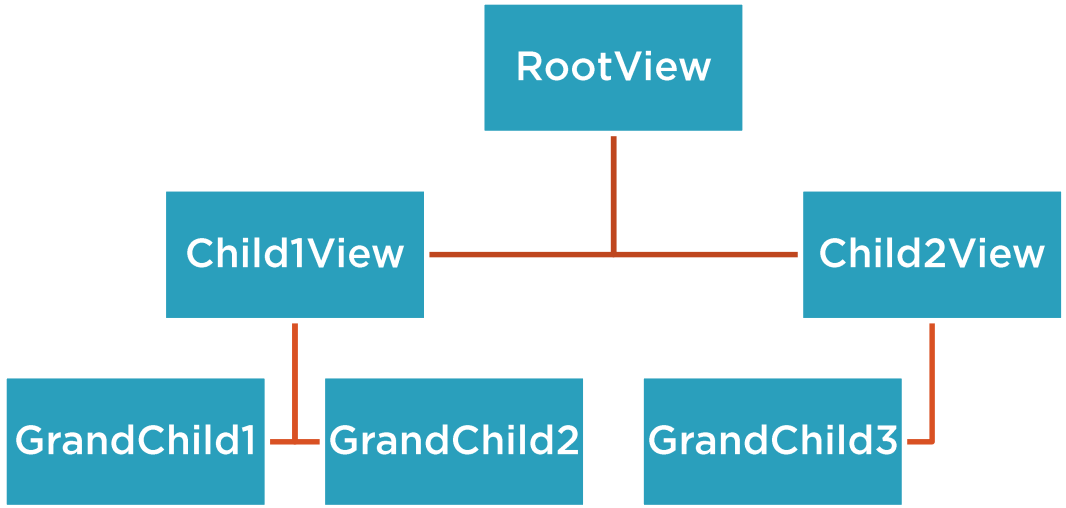
Nested content may or may not need to be Views with ViewModels

If no encapsulated data management or interaction logic, no need for a ViewModel

Parent ViewModels can construct child ViewModels

And cause navigation, or view switching through DataTemplates

Parent ViewModels can supervise and mediate between children



Applied MVVM Demo Use Cases

List customers

Place orders for a customer

Add/edit customers

Monitor order preparation

Summary

Naming and location of components is important for maintainability

Hierarchical MVVM mirrors the way we normally compose complex screens

Building out an MVVM app is a progressive sequence of defining Views, ViewModels, navigation and communications to satisfy the requirements