

WPF MVVM In Depth

MVVM Pattern Fundamentals



Brian Noyes

@briannoyes | www.solliance.net

Model-View-ViewModel (MVVM)
is mostly about
trying to achieve good
Separation of Concerns

No Separation of Concerns



Easy to put clothes away

Really hard to get dressed

Good Separation of Concerns

A bit more work to put things away

Makes getting dressed easy



No Separation of Concerns

```
private void ComputeCustomerOrdersTotal(object sender, RoutedEventArgs e)
{
    var selectedCustomer = this.customerDataGrid.SelectedItem as Customer;
    var orders = (from order in dbContext.Orders.Include("OrderItems")
                  where order.CustomerId == selectedCustomer.Id
                  select order);

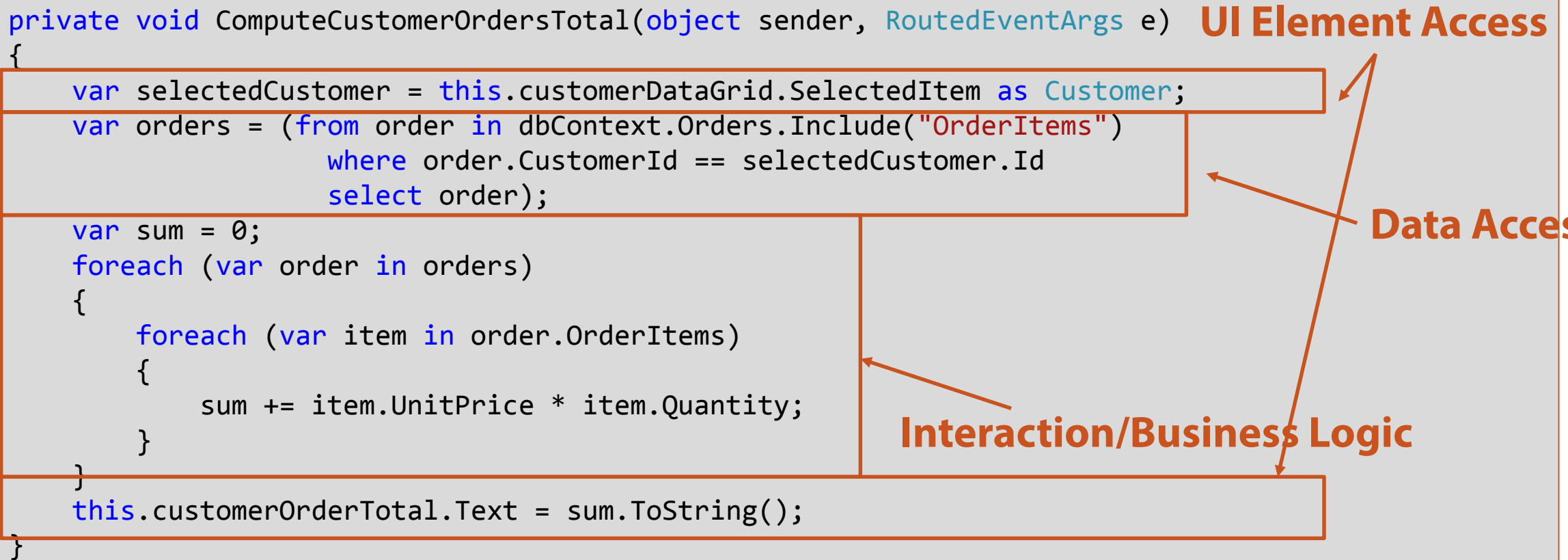
    var sum = 0;
    foreach (var order in orders)
    {
        foreach (var item in order.OrderItems)
        {
            sum += item.UnitPrice * item.Quantity;
        }
    }

    this.customerOrderTotal.Text = sum.ToString();
}
```

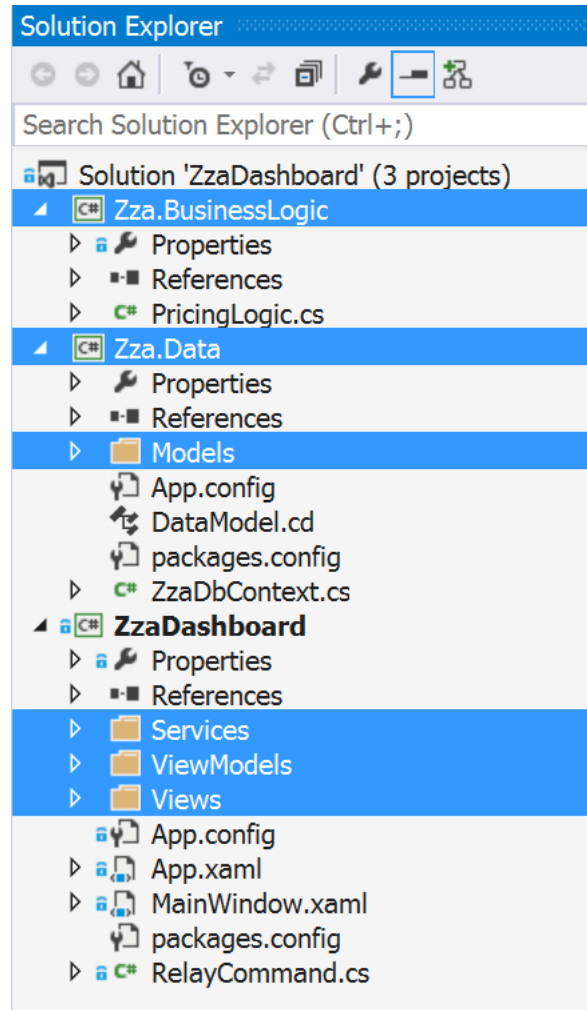
UI Element Access

Data Access

Interaction/Business Logic



Good Separation of Concerns



← **Business Logic**

← **Data Access**

← **Model Entities**

← **Shared Client Logic**

← **View Interaction Logic**

← **UI Element Access**

MVVM Goals / Benefits

Maintainability

Testability

Extensibility

Model-View-Controller (MVC)



Dates back to early 1970's

Favored by modern Web platforms

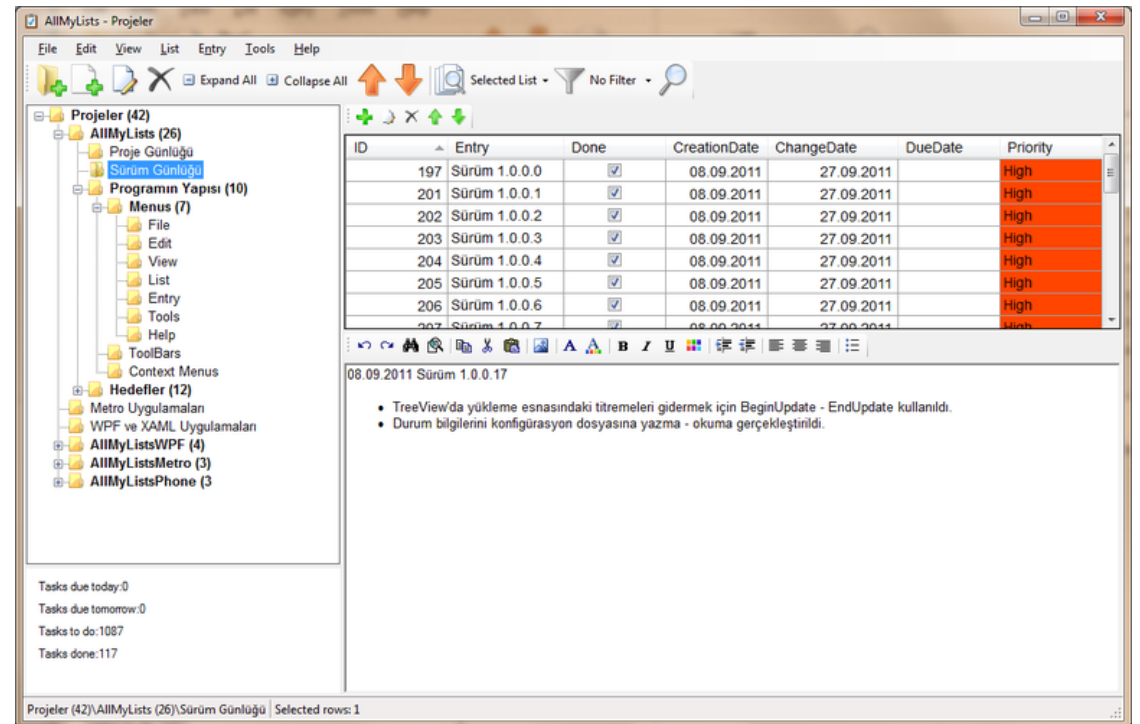
Lifetime separation between
Controller and View

Model-View-Presenter (MVP)

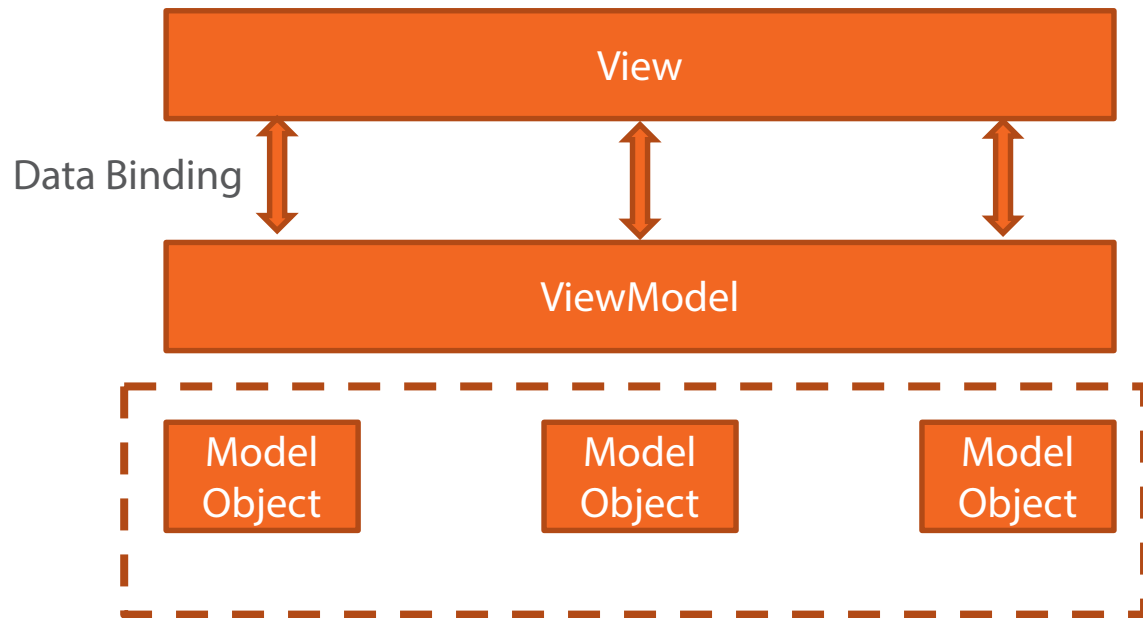
Mid-2000's Rich / Smart Client Apps

Ongoing communication between
View and Presenter

Primarily method calls back and
forth – interface decoupling



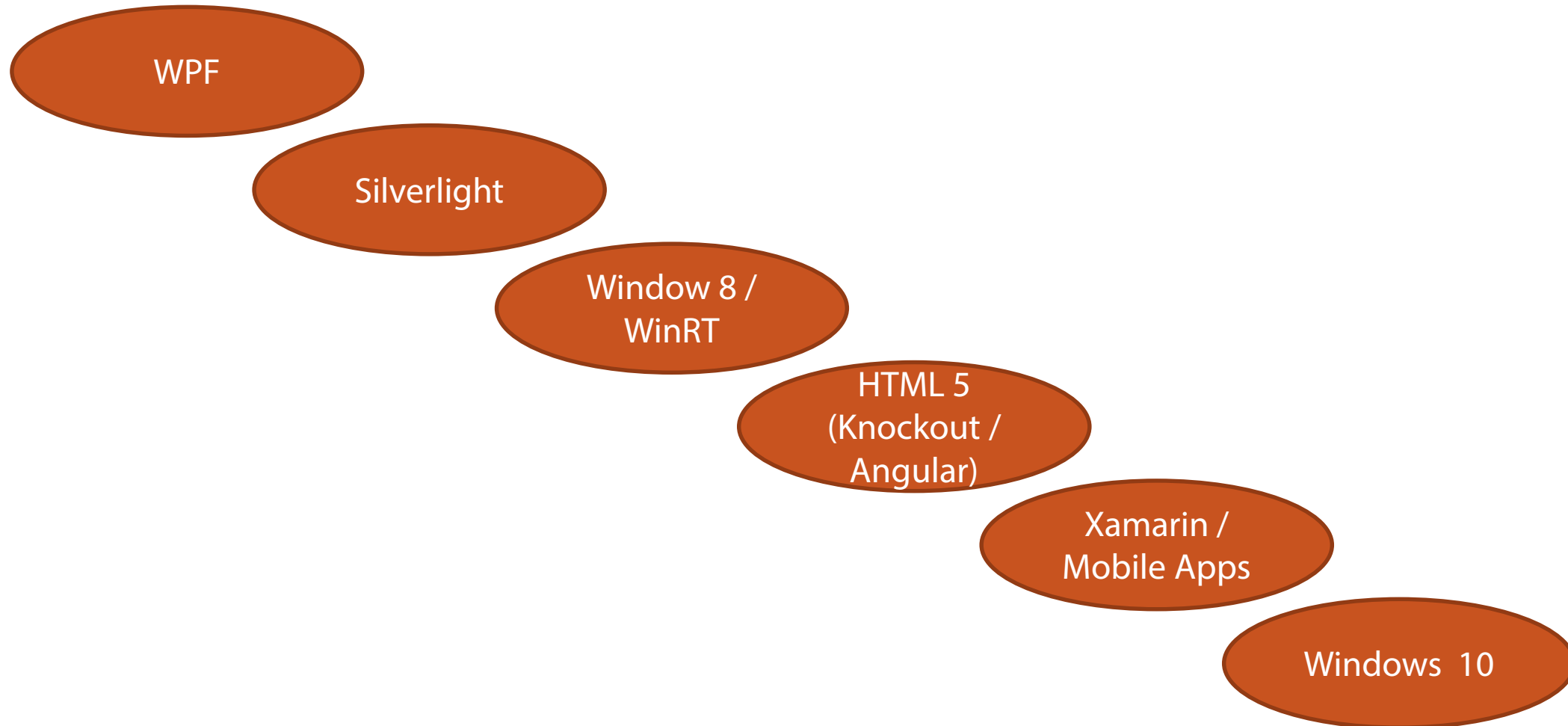
Model-View-ViewModel (MVVM)



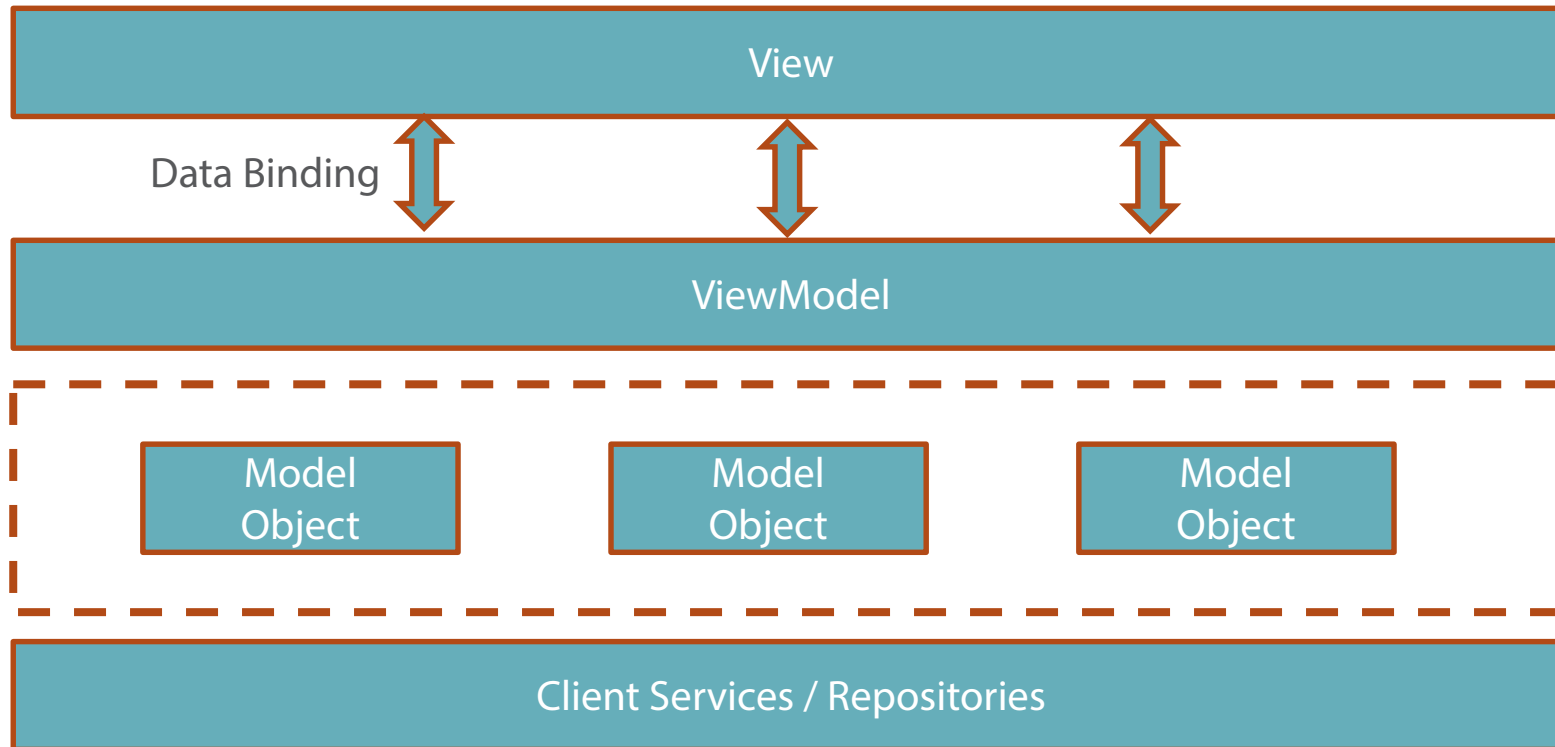
Ongoing interaction between View and ViewModel

Data flow and communications primarily through data binding

MVVM Across Platforms

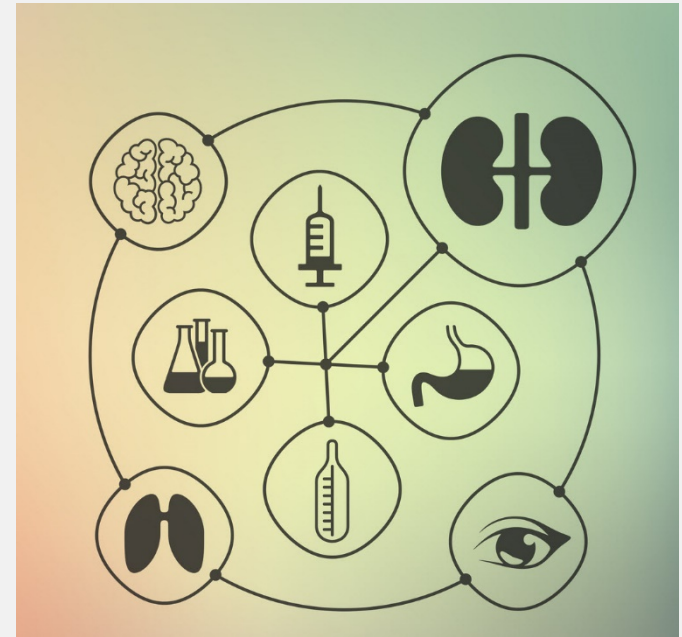


MVVM Responsibilities



Model Responsibilities

- Contain the client data
- Expose relationships between model objects
- Computed properties
- Raise change notifications
 - INotifyPropertyChanged.PropertyChanged
- Validation
 - INotifyDataErrorInfo/IDataErrorInfo



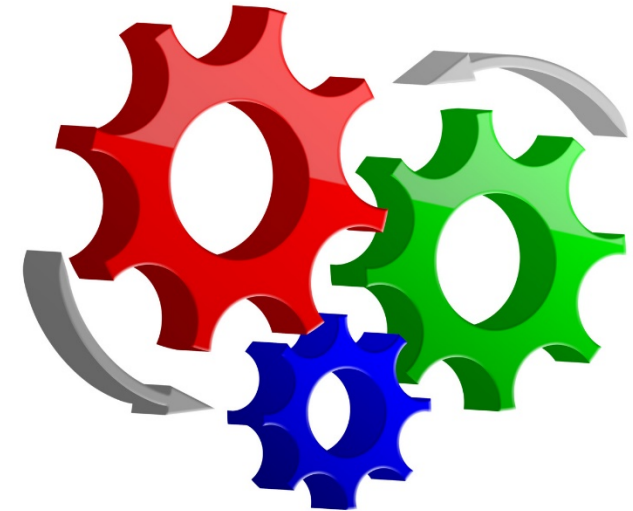
View Responsibilities

- Structural definition of what the user sees on the screen
- GOAL: “No Code Behind”
- Reality: Sometimes need some code behind

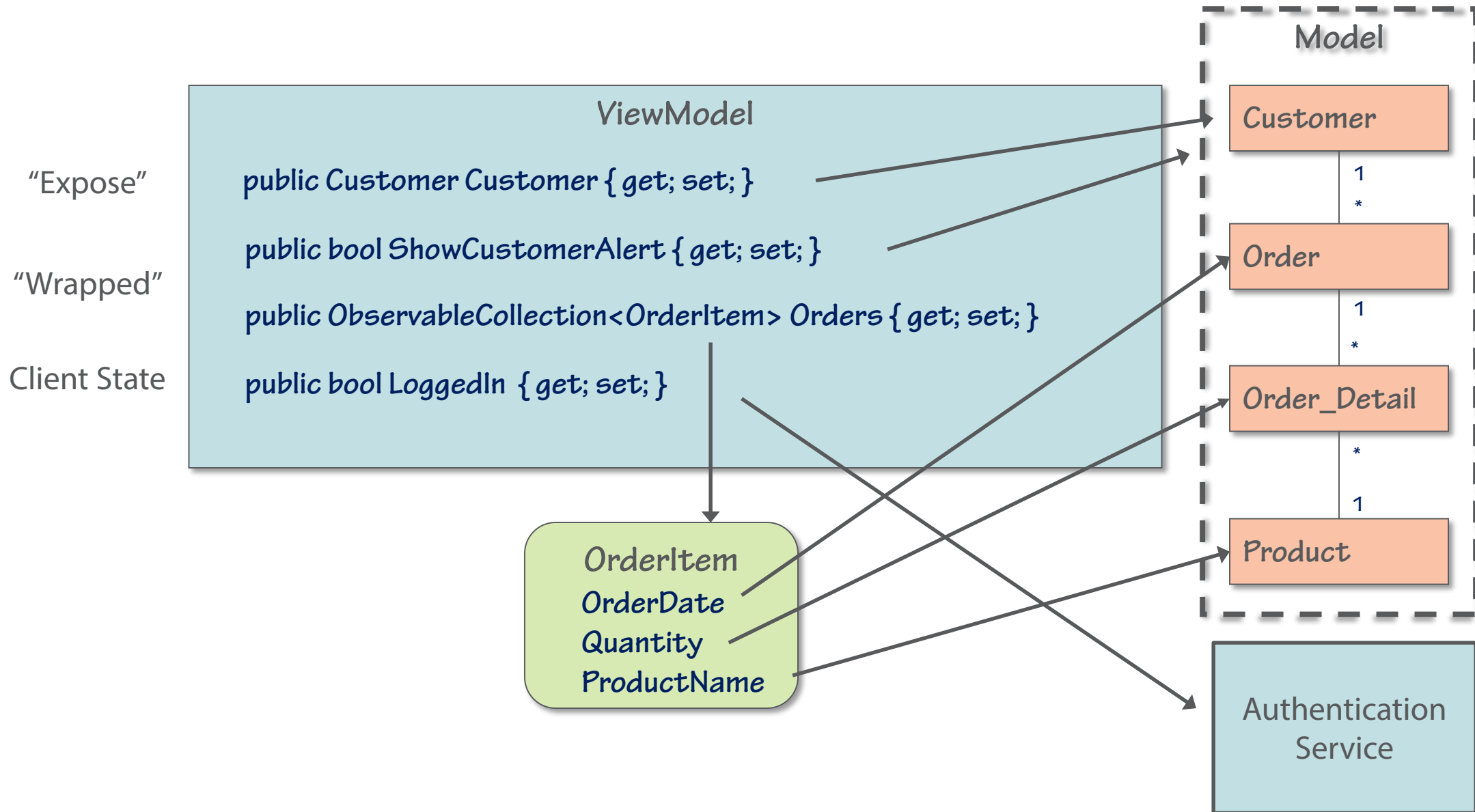


ViewModel Responsibilities

- Expose data to the view for presentation and manipulation
- Encapsulate interaction logic
 - Calls to business/data layer/service
 - Navigation logic
 - State transformation logic



ViewModel Data

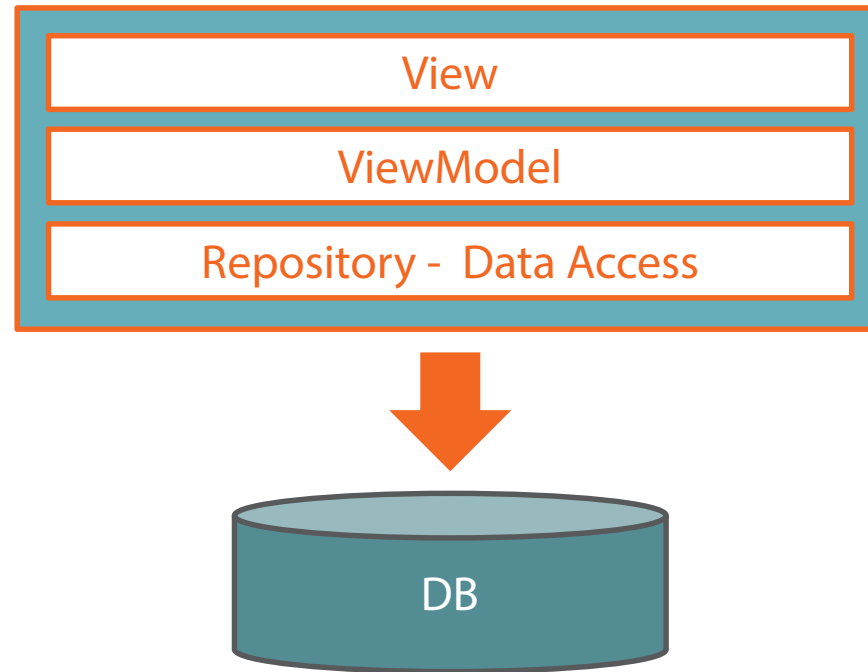


Client Services / Repositories

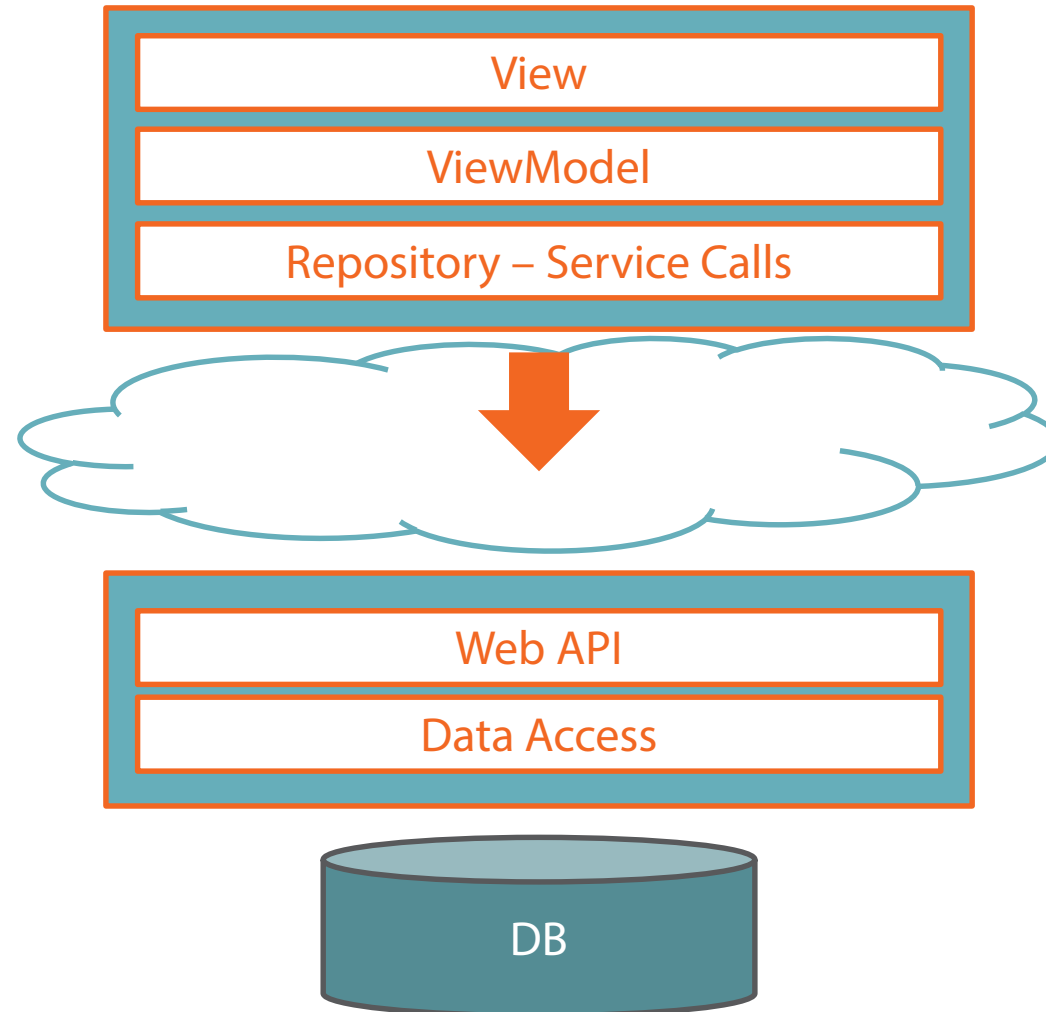
- Shared functionality or data access
- Consumed by one or more ViewModels
- Decouples ViewModels from external dependencies
 - Data storage
 - Service access
 - Client environment
- Can act as data caching container



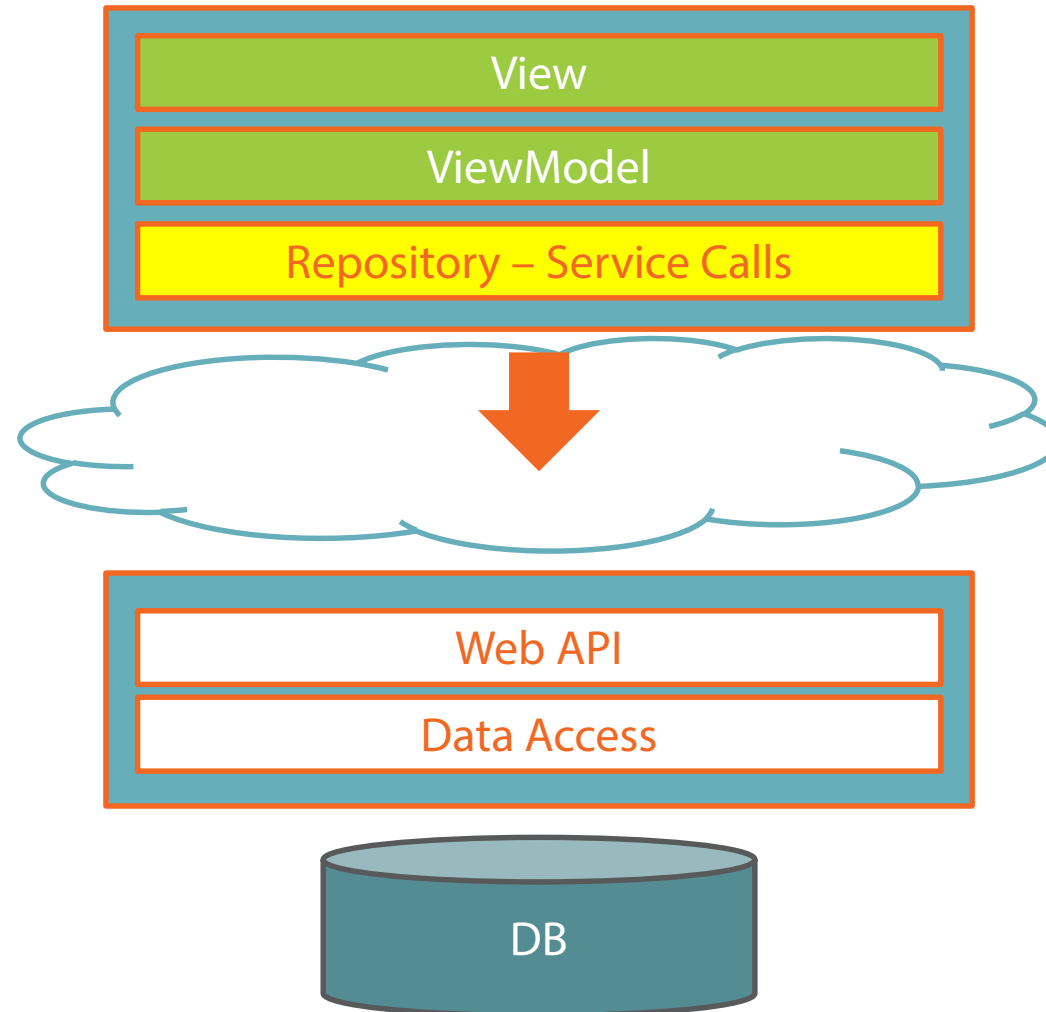
Client Services / Repositories



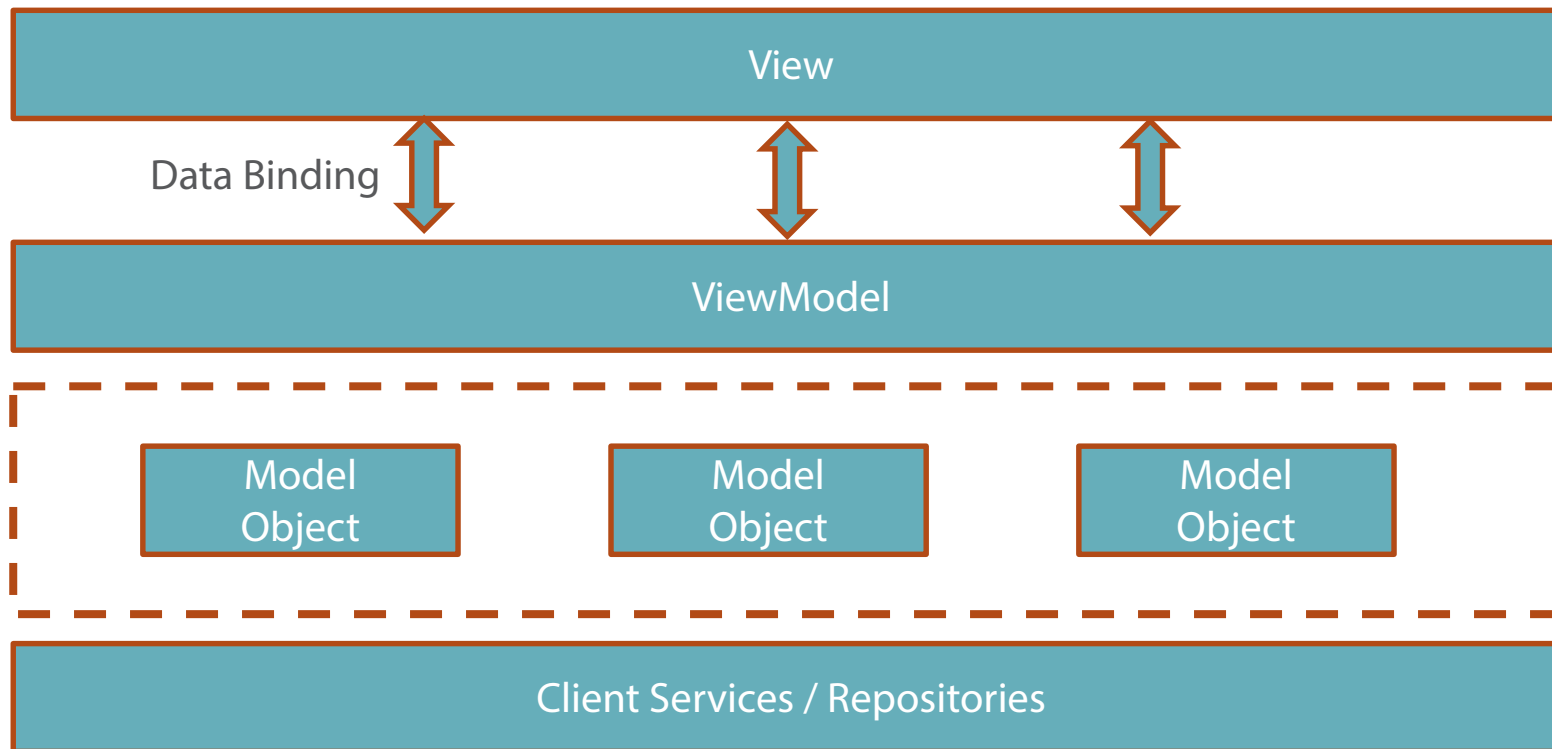
Client Services / Repositories



Client Services / Repositories



MVVM Responsibilities



Fundamental equation of MVVM:

View.DataContext = ViewModel

View-ViewModel Instantiation

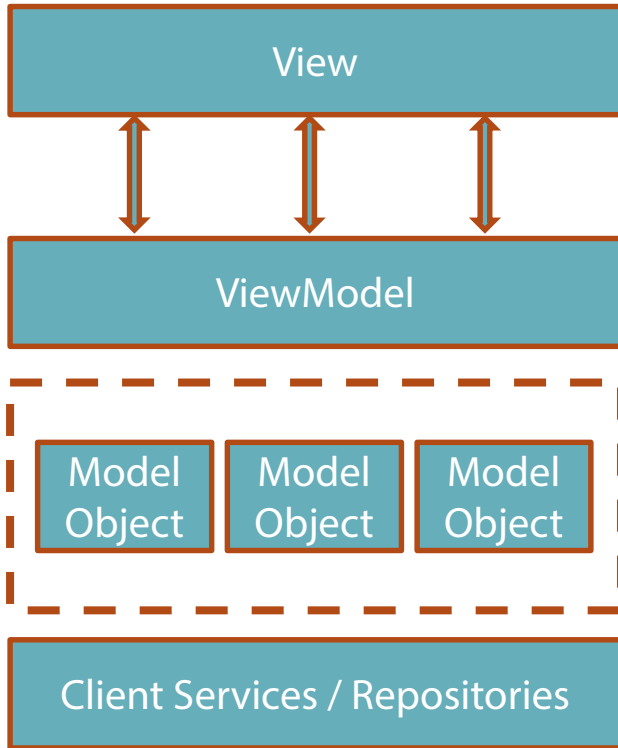
View-First

- View is constructed first
- ViewModel gets constructed and attached to DataContext via View

ViewModel-First

- ViewModel is constructed first
- View is constructed as a consequence of ViewModel being added to UI

MVVM Concepts Summary



MVVM helps you build better structured apps

Each part has a specific responsibility

`View.DataContext = ViewModel`