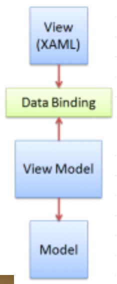
M-V-VM Model

Thursday, June 21, 2018 9:19 AM

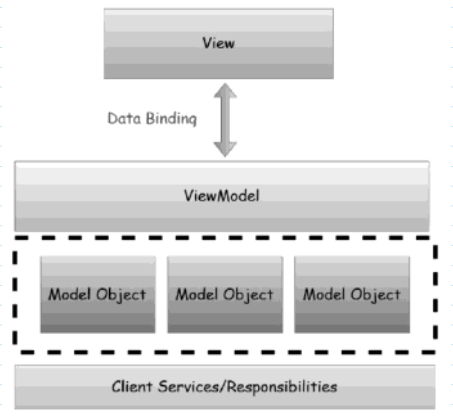


Model View ViewModel - helps separate concerts to facilitate testability by enabling to isolate concerns
- Model: Layer that represents the application's data
  - Domain object, or the actual data/information we deal with -> the characteristics of something
  - Holds the information, but not behaviors or services that manipulate the information - doesn't deal with making stuff on the screen
  - Ex. If we had a Contact info mode, we might provide first names, last names, full name, phone numbers
- View: The presentation or the user interface layer/formats data
  - he only thing the end user interacts with, essentially the presentation of the data
- Controller/ViewModel: Layer that contains business logic of the application - introduces presentation separation to keep nuances of the view separate from the model
  - Keeps nuances of the view separate from the model (converts the date to a display format)
  - Acts as the liaison between the model (data) and the view (holds it) by taking input from the view and putting it in the mode l or interact with a service to retrieve a model an translate properties to put it in the view

Separation Presentation: Avoid this: where there is Xaml and backend code with minimum required for working with the UI direc tly

Advantages:
- True separation between the view and model
- Maintaining is easy because of agile parts
- Testability is efficient
- Extendable because we can replace or add new pieces that do similar things in the right places

Responsibilities of MVVM

- 

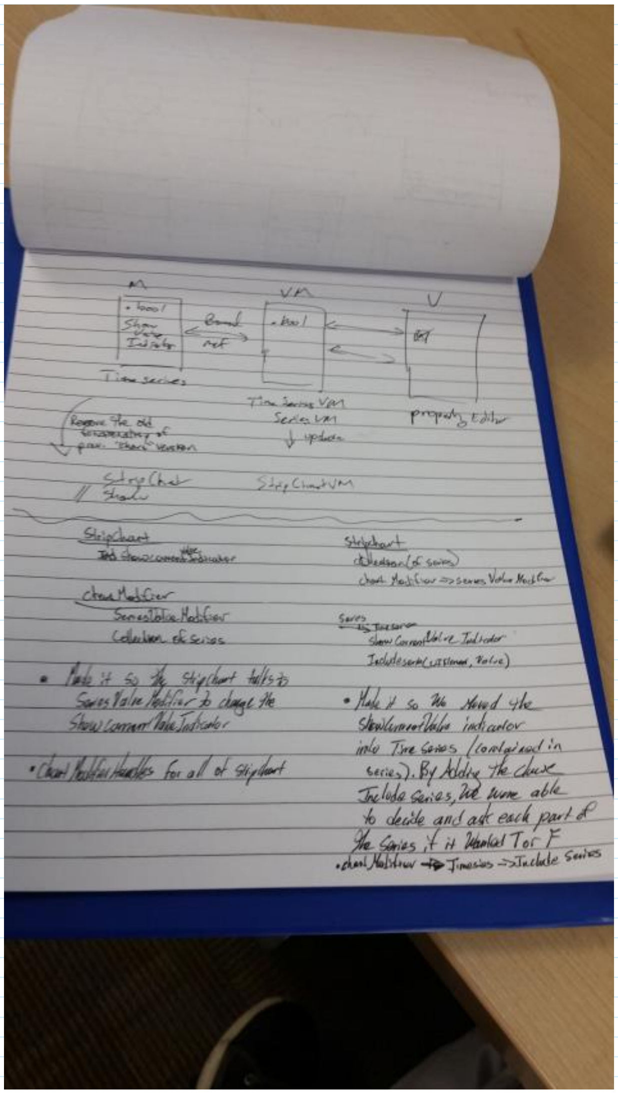
- Model:
  - Supports the views in the applications, composed of objects with properties
    - Can reference other model objects and create object graph
  - Validation of databinding
- View
  - Define what the user sees on the screen (dynamic and staic)
    - Static parts define controls and layout of controls
    - Dynamic includes animations or state changes
  - No code behind in the view - need at least constructor and call to initialize
  - No Event Handling
- ViewModel
  - Provide data to view and allows users to interact with data and change data
  - Handle sequencing of calls and manage any navigation logic to when to navigate to a different view

View and ViewModel
- Communicate via data binding, method calls, properties, events and messages
- Viewmodel exposes not only to models but other functions like "is busy" indicators and commands
- View handles own UI events and maps them to the viewmodel
- Model and properties on the viewmodel are updated through 2 way databinding

ViewModel and Model
- ViewModel becomes responsible for the model
- Viewmodel may expose the model directly or properties related to the model for databinding
- ViewModel can contain interfaces to services, configuration data to see what properties to expose to the view

Conventionally design the view before the viewmodel, but either way works

Usually have 1 view per viewmodel


M
• bool
Show Value Indicator
Bound
ref
VM
• bool
V
Time series
Time Series VM
Series VM
↓ update
property Editor
Remove the old [illegible] of prev. "Share" version
StripChart
// Show
StripChartVM

StripChart
Show current Value Indicator

chart Modifier
SeriesValue Modifier
Collection of series

• Make it so the StripChart talks to SeriesValue Modifier to change the Show current Value Indicator

• Chart Modifier Handles for all of StripChart

StripChart
collection (of series)
chart Modifier => series Value Modifier

Series
-> TimeSeries
Show Current Value Indicator
IncludeSeries(UIElement, Value)

• Make it so we Moved the ShowCurrentValue indicator into Time series (contained in series). By Adding the class Include series, we were able to decide and ask each part of the series if it Wanted T or F

• chart Modifier -> Timeseries -> Include Series