Enterprise MVVM Document

What is MVVM? It’s a design pattern to separate the view, viewmodel, and model. It’s usually used in WPF and Silverlight projects because of the data binding they provide.

Overview:

* Uses .Net 4.5, Visual studio 2015
* Test Driven Development
* Create ObservableObject class for change notification
* Create Abstract base View Model class
* Utilize Observable Object
* Utilize IDataErrorInfo interface
* Utilize Data Annotations
* Derived viewmodel’s can override base implementation

Video 1:

1. Create a base ViewModel, so other ViewModel like CustomerViewModel and inherit so you don’t have to implement IDataErrorInfo and other implementation that’s common in each all ViewModel to reduce duplicate codes. This way it’s easier to maintain.
2. Create a new Visual Studio Solution (In “Other Project Type”)
   1. Call it EnterpriseMVVM
3. Create and add a C# Windows WPF Application
   1. Call it EnterpriseMVVM.DesktopClient
4. Create and add a C# Windows Class Library
   1. Call it EnterpriseMVVM.Windows
   2. Delete the default Class.cs
5. Create and add a C# Test Unit Test Project
   1. Call it “Test”
   2. Delete the default UnitTest.cs
6. Create 2 new folder in EnterpriseMVVM.DesktopClient
   1. Views
   2. ViewModels
7. Update the namespaces for the MainWindow.xaml amd MainWindow.xaml.cs
   1. Move the MainWindow.xaml into the “Views” folder
   2. “EnterpriseMVVM.DesktopClient.Views.MainWindow
8. In Test project add a “ObservableObjectTests.cs class
   1. Create a “PropertyChangeEventHandlerIsRaised()
   2. Add a reference to the “Tests” unit
      1. EnterpriseMVVM.Windows
9. Create the StubObservableObject.cs class in the EnterpriseMVVM.Windows project
   1. Implement the INotifyPropertyChanged
      1. Use the System.RunTime.ComplierService
         1. To skip passing in a value and the compiler will pass in the property object name as the parameter