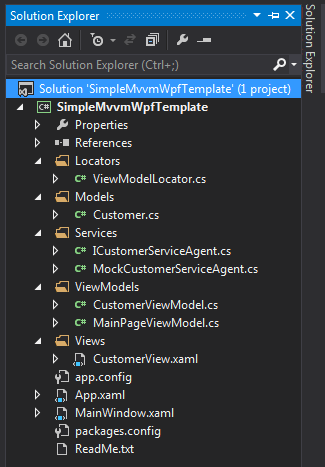
**Template project Simple MVVM toolkit**

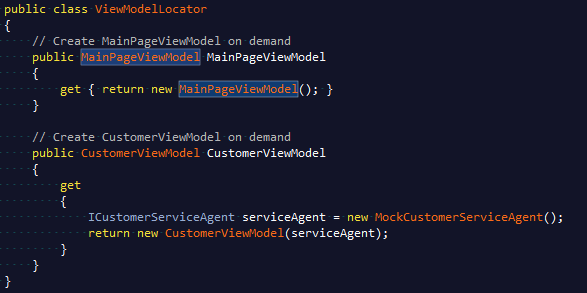
Cấu trúc Project:



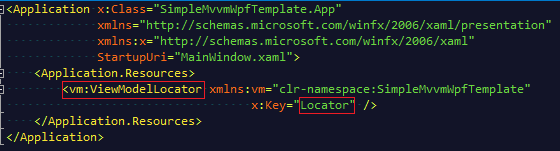
Trong Project được phân chia khá rõ rang cho việc phân tách các chức năng:

* Folder Locator.
* Folder Models.
* Folder Services.
* Folder ViewModels
* Folder Views.

1. **Folder Locator** : chứa class ViewModelLocator chứa thuộc tính chỉ đọc trả về giá trị là các class ViewModel để View có thể binding đến mà không cần phải Binding trực tiếp từ class.



Khai báo class ViewModelLocator này với key là "Locator"  trong file App.xaml chứa Resource của Project.



Sau cùng trong view CustomerView chỉ cần khai báo thuộc tính DataContext của Control mẹ và binding đến source có key "Locator" đã khai báo trong Resources trước đó và khai báo thêm thuộc tính đường dẫn thuộc tính Path = "<tên của thuộc tính trả về class ViewModel cần binding>"

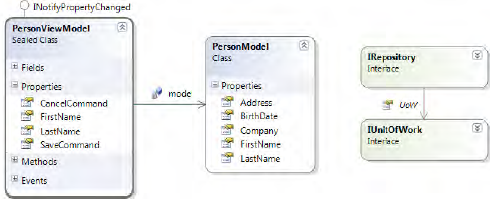


1. **Folder Models:** Có chứa các class model.

Model là 1 Domain Entity , không nên nhầm lẫn Model và ViewModel vì ViewModel mới tiếp xúc với View còn Model thì không.

Xa hơn nữa Model là entity chịu trách nhiệm di chuyển dữ liệu đến và đi từ datastore bao gồm các đối tượng được biết đến như là DataLayer & BusinessLayer.

Một ví dụ về giao tiếp giữa ViewModel và Model



- View nhận biết và theo dõi ViewModel nhưng không biết Model

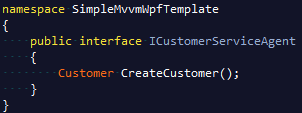
- ViewModel thì nhận biết được Model có nhiệm update dữ liệu từ Model lên View

- Model thì không biết View cũng như ViewModel

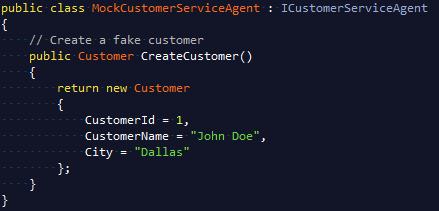
- DAL có nhiệm vụ là save dữ liệu từ Model xuống Database.

1. **Folder Services:** chứa các class tạo phương thức control với dữ liệu được khai báo trong Model.

**Ví dụ:** với class Customer ta tạo 2 service:

****

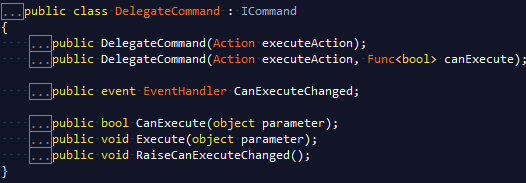
Và:



Để thao tác đối với dữ liệu ở Model.

1. **Folder ViewModel:** chứa các class giao tiếp với Model, control dữ liệu tới View.

Trong Project mẫu xây dựng bằng Simple MVVM toolkit này **Class DelegateCommand dùng để hiện thực hóa các sự kiện ở View sang ViewModel sử dụng Interface ICommand** đã được sinh ra sẵn.



Gồm các phương thức CanExecute, Execute, RaiseCanExecuteChanged.

- ICommand hiện thực 2 phương thức Execute và CanExecute cho phép bạn điều khiển việc thực hiện -lệnh.

- Bằng cách sử dụng Binding và hiện thực ICommand bạn có thể code 1 ViewModel thực hiện nhiều lệnh nhận được từ View và binding đến Control trên View như Button , Link ...

- IComand cho phép bạn điều khiển lệnh dựa trên những điều kiện thay đổi trên ViewModel ví dụ như chỉ có thể thực hiện lệnh Save trên View với điều kiện trên ViewModel đã thực hiện OnPropertyChanged một lần nghĩa là đã edit trên view.

Thông thường bạn cần trình bày các command trong ViewModel như là các thuộc tính public để tạo ràng buộc giữa View & ViewModel.Trình bày 1 thuộc tính Icommand công cộng trong ViewModel cho phép View có thể binding đến chúng.Bạn có thể hiện thực ICommand với nhiều cách và cũng cần hiện thực một số presentation logic trong ViewModel để quyết định lệnh có thể không không thể thực hiện.

- ViewModel phải đáp ứng 4 yêu cầu sau

+ Cung cấp dữ liệu hiển thị lên View

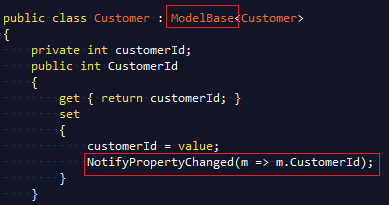
+ Cung cấp bộ lệnh cho View

+ Hiện thực interface INotifyPropertyChanged

+ Hiện thực interface IDataErrorInfo

**INotifyPropertyChanged**

Trong khung project này thì phương thức INotifyPropertyChanged đã được sinh ra sẵn và được sử dụng trong model khi khai báo một cách dễ dàng.



**IDataErrorInfo**

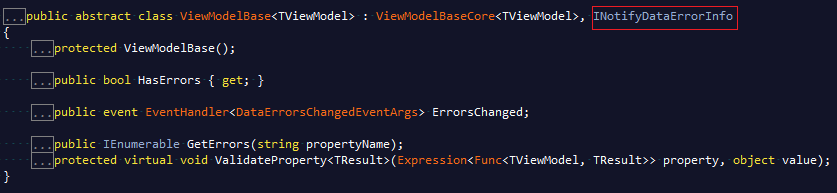
 Interface IDataErrorInfo cũng nằm trong namespace System.ComponentModel.Nó được xây dựng nhằm mục đích cung cấp thông tin lỗi cụ thể cho 1 đối tượng binding với giao diện người dùng hay trình khách (View)

- IDataErrorInfo có 2 thuộc tính : Error và Item.

+ Thuộc tính Error trình bày lỗi xác nhận hiện tại. Đây là hiện thực thông thường nhất cho ứng dụng và bạn cũng có thể sử dụng XAML DataTemplate để hiển thị Validation Error.

+ Thuộc tính Item được gọi mỗi khi 1 element trên View xác nhận cho phép gọi đến validation engine, validation engine này thay đổi các giá trị của nó hoặc yêu cầu xác nhận giá trị của nó.

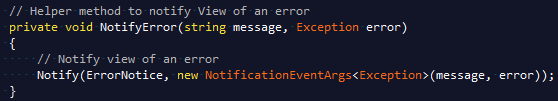
Trong Project này, phương thức IDataErrorInfo đã được tạo sẵn trong ViewModelBass



Khi sử dụng chỉ cần khai báo trong ViewModel:



Và thực hiện đưa lỗi ra View qua hàm:



Đối với các công cụ hỗ trợ khác, cũng xây dựng chủ với những cấu trúc như trên. Chỉ khác nhau ở chỗ, các phần chính chia ra khác nhau, và các phần code viết sẵn được thể hiện ra rõ ràng hoặc được ẩn đi vào bên trong.

Còn về khung chính thì đều như trên. Có thể phân chia nhỏ hơn để dễ dàng hơn trong quá trình quản lý và khai thác project.