This article describes the basic use and functionality of the MVVM pattern in WPF.  
  
[The Model View ViewModel (MVVM)](https://www.c-sharpcorner.com/uploadfile/nipuntomar/mvvm-in-wpf/) is an architectural pattern used in software engineering that originated from Microsoft which is specialized in the Presentation Model design pattern. It is based on the Model-view-controller pattern (MVC), and is targeted at modern UI development platforms (WPF and Silverlight) in which there is a UX developer who has different requirements than a more "traditional" developer. MVVM is a way of creating client applications that leverages core features of the WPF platform, allows for simple unit testing of application functionality, and helps developers and designers work together with less technical difficulties.  
  
**VIEW:** A View is defined in XAML and should not have any logic in the code-behind. It binds to the view-model by only using data binding.  
  
**MODEL:**AModel is responsible for exposing data in a way that is easily consumable by WPF. It must implement INotifyPropertyChanged and/or INotifyCollectionChanged as appropriate.  
  
**VIEWMODEL:** A ViewModel is a model for a view in the application or we can say as abstraction of the view. It exposes data relevant to the view and exposes the behaviors for the views, usually with Commands.  
  
**Getting Started**

* Creating a WPF Project. Open Visual Studio2010.
* Go to File => New => Project
* Select Window in installed templates
* Select WPF Application
* Enter the Name and choose the location.
* Click OK

Now create three folders in root application. Name should be Model,View,ViewModel and now add a new class in Model folder. My class name is User and add this namespace

1. **using** System.ComponentModel;

**User.cs**

1. **public** **class** User : INotifyPropertyChanged
2. {
3. **private** **int** userId;
4. **private** **string** firstName;
5. **private** **string** lastName;
6. **private** **string** city;
7. **private** **string** state;
8. **private** **string** country;
9. **public** **int** UserId
10. {
11. **get**
12. {
13. **return** userId;
14. }
15. **set**
16. {
17. userId = value;
18. OnPropertyChanged("UserId");
19. }
20. }
21. **public** **string** FirstName
22. {
23. **get**
24. {
25. **return** firstName;
26. }
27. **set**
28. {
29. firstName = value;
30. OnPropertyChanged("FirstName");
31. }
32. }
33. **public** **string** LastName
34. {
35. **get**
36. {
37. **return** lastName;
38. }
39. **set**
40. {
41. lastName = value;
42. OnPropertyChanged("LastName");
43. }
44. }
45. **public** **string** City
46. {
47. **get**
48. {
49. **return** city;
50. }
51. **set**
52. {
53. city = value;
54. OnPropertyChanged("City");
55. }
56. }
57. **public** **string** State
58. {
59. **get**
60. {
61. **return** state;
62. }
63. **set**
64. {
65. state = value;
66. OnPropertyChanged("State");
67. }
68. }
69. **public** **string** Country
70. {
71. **get**
72. {
73. **return** country;
74. }
75. **set**
76. {
77. country = value;
78. OnPropertyChanged("Country");
79. }
80. }
82. #region INotifyPropertyChanged Members
84. **public** **event** PropertyChangedEventHandler PropertyChanged;
85. **private** **void** OnPropertyChanged(**string** propertyName)
86. {
87. **if** (PropertyChanged != **null**)
88. {
89. PropertyChanged(**this**, **new** PropertyChangedEventArgs(propertyName));
90. }
91. }
92. #endregion
94. }

Now right click on ViewModel folder and add a new class.

**UserViewModel.cs**

1. **using** System.Windows.Input;
2. **using** System.ComponentModel;
4. **class** UserViewModel
5. {
6. **private** IList<User> \_UsersList;
8. **public** UserViewModel()
9. {
10. \_UsersList = **new** List<User>
11. {
12. **new** User{UserId = 1,FirstName="Raj",LastName="Beniwal",City="Delhi",State="DEL",Country="INDIA"},
13. **new** User{UserId=2,FirstName="Mark",LastName="henry",City="New York", State="NY", Country="USA"},
14. **new** User{UserId=3,FirstName="Mahesh",LastName="Chand",City="Philadelphia", State="PHL", Country="USA"},
15. **new** User{UserId=4,FirstName="Vikash",LastName="Nanda",City="Noida", State="UP", Country="INDIA"},
16. **new** User{UserId=5,FirstName="Harsh",LastName="Kumar",City="Ghaziabad", State="UP", Country="INDIA"},
17. **new** User{UserId=6,FirstName="Reetesh",LastName="Tomar",City="Mumbai", State="MP", Country="INDIA"},
18. **new** User{UserId=7,FirstName="Deven",LastName="Verma",City="Palwal", State="HP", Country="INDIA"},
19. **new** User{UserId=8,FirstName="Ravi",LastName="Taneja",City="Delhi", State="DEL", Country="INDIA"}
20. };
21. }
23. **public** IList<User> Users
24. {
25. **get** { **return** \_UsersList; }
26. **set** { \_UsersList = value; }
27. }
29. **private** ICommand mUpdater;
30. **public** ICommand UpdateCommand
31. {
32. **get**
33. {
34. **if** (mUpdater == **null**)
35. mUpdater = **new** Updater();
36. **return** mUpdater;
37. }
38. **set**
39. {
40. mUpdater = value;
41. }
42. }
44. **private** **class** Updater : ICommand
45. {
46. #region ICommand Members
48. **public** **bool** CanExecute(**object** parameter)
49. {
50. **return** **true**;
51. }
53. **public** **event** EventHandler CanExecuteChanged;
55. **public** **void** Execute(**object** parameter)
56. {
58. }
60. #endregion
61. }
62. }

Now let's move on View, Add a new windows in View Folder.

**MainPage.xaml**

1. <Window x:Class="WpfMVVMSample.MainWindow"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. Title="MainWindow" Height="485" Width="525">
5. <Grid Margin="0,0,0,20">
6. <Grid.RowDefinitions>
7. <RowDefinition Height="Auto"/>
8. <RowDefinition Height="\*"/>
9. <RowDefinition Height="Auto"/>
10. </Grid.RowDefinitions>
11. <ListView Name="UserGrid" Grid.Row="1" Margin="4,178,12,13"  ItemsSource="{Binding Users}"  >
12. <ListView.View>
13. <GridView x:Name="grdTest">
14. <GridViewColumn Header="UserId" DisplayMemberBinding="{Binding UserId}"  Width="50"/>
15. <GridViewColumn Header="First Name" DisplayMemberBinding="{Binding FirstName}"  Width="80" />
16. <GridViewColumn Header="Last Name" DisplayMemberBinding="{Binding LastName}" Width="100" />
17. <GridViewColumn Header="City" DisplayMemberBinding="{Binding City}" Width="80" />
18. <GridViewColumn Header="State" DisplayMemberBinding="{Binding State}" Width="80" />
19. <GridViewColumn Header="Country" DisplayMemberBinding="{Binding Country}" Width="100" />
20. </GridView>
21. </ListView.View>
22. </ListView>
23. <TextBox Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="80,7,0,0" Name="txtUserId" VerticalAlignment="Top" Width="178" Text="{Binding ElementName=UserGrid,Path=SelectedItem.UserId}" />
24. <TextBox Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="80,35,0,0" Name="txtFirstName" VerticalAlignment="Top" Width="178" Text="{Binding ElementName=UserGrid,Path=SelectedItem.FirstName}" />
25. <TextBox Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="80,62,0,0" Name="txtLastName" VerticalAlignment="Top" Width="178" Text="{Binding ElementName=UserGrid,Path=SelectedItem.LastName}" />
26. <Label Content="UserId" Grid.Row="1" HorizontalAlignment="Left" Margin="12,12,0,274" Name="label1" />
27. <Label Content="Last Name" Grid.Row="1" Height="28" HorizontalAlignment="Left" Margin="12,60,0,0" Name="label2" VerticalAlignment="Top" />
28. <Label Content="First Name" Grid.Row="1" Height="28" HorizontalAlignment="Left" Margin="12,35,0,0" Name="label3" VerticalAlignment="Top" />
29. <Button Content="Update" Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="310,40,0,0" Name="btnUpdate"
30. VerticalAlignment="Top" Width="141"
31. Command="{Binding Path=UpdateCommad}"  />
32. <TextBox Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="80,143,0,0" x:Name="txtCity" VerticalAlignment="Top" Width="178" Text="{Binding SelectedItem.City, ElementName=UserGrid}" />
33. <Label Content="Country" Grid.Row="1" Height="28" HorizontalAlignment="Left" Margin="12,141,0,0" x:Name="label2\_Copy" VerticalAlignment="Top" />
34. <TextBox Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="80,88,0,0" x:Name="txtCountry" VerticalAlignment="Top" Width="178" Text="{Binding SelectedItem.Country, ElementName=UserGrid}" />
35. <Label Content="City" Grid.Row="1" Height="28" HorizontalAlignment="Left" Margin="12,86,0,0" x:Name="label2\_Copy1" VerticalAlignment="Top" />
36. <TextBox Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="80,115,0,0" x:Name="txtSTate" VerticalAlignment="Top" Width="178" Text="{Binding SelectedItem.State, ElementName=UserGrid}" />
37. <Label Content="State" Grid.Row="1" Height="28" HorizontalAlignment="Left" Margin="12,113,0,0" x:Name="label2\_Copy2" VerticalAlignment="Top" />
38. </Grid>
39. </Window>

**App.xaml.cs**  
  
Now bind on application startup.

1. **protected** **override** **void** OnStartup(StartupEventArgs e)
2. {
3. **base**.OnStartup(e);
4. WpfMVVMSample.MainWindow window = **new** MainWindow();
5. UserViewModel VM = **new** UserViewModel();
6. window.DataContext = VM;
7. window.Show();
8. }

Run the application to see the result.

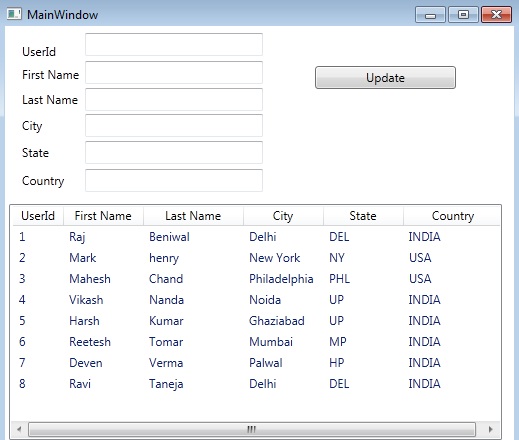


Image 1

Click on row.

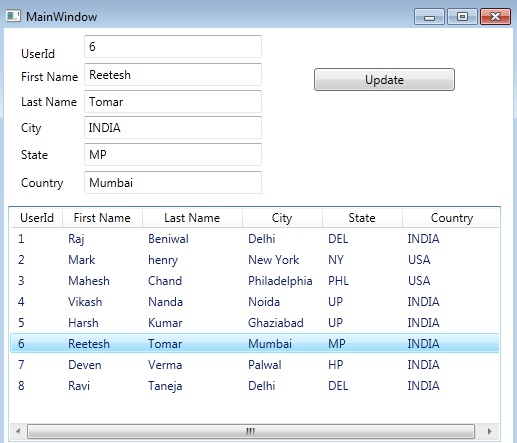


Image 2

You can edit user detail in textboxes and click update button.

