Silverlight 4

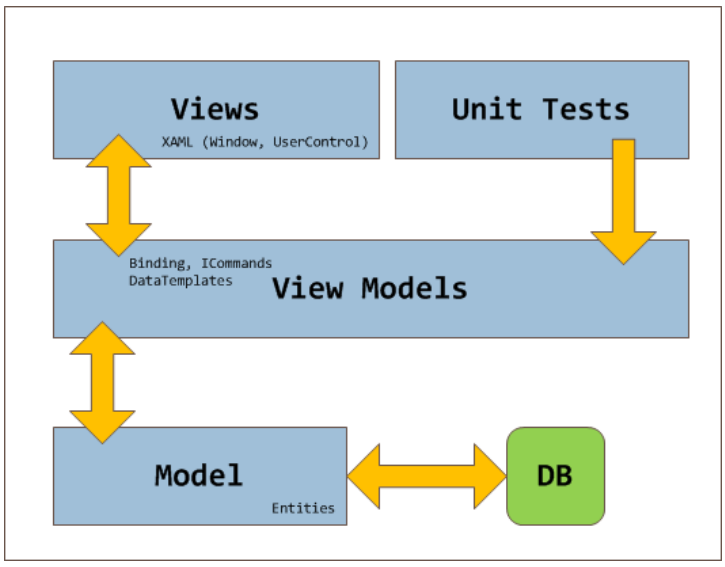
پژوهش و نگارش: وحید نصیری
Vahid_Nasiri@Yahoo.Com
بهار ۱۳۸۹

فهرست مطالب

فصل ۱۰ - معرفی مثالی مقدماتی از پیاده سازی الگوی M-V-VM در Silverlight

مقدمه

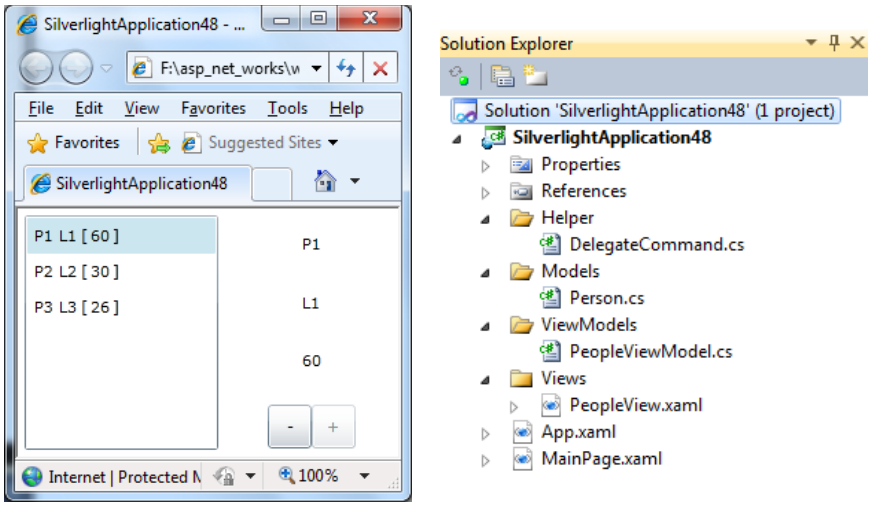
هر چند اکثر مثال‌های فصل معرفی سیستم Binding در Silverlight بر اساس الگوی M-V-VM پیاده سازی شدند، اما اکنون پس از مطالعه‌ی تئوری این الگو و بایدها و نبایدهای آن، بهتر می‌توان به درک و پیاده سازی اصولی آن پرداخت. همچنین باتوجه به پیشرفت‌های اخیر Silverlight 4 در مورد کار با اشیاء Command ، استفاده از الگوی M-V-VM در Silverlight به سادگی پیاده سازی آن در دنیای WPF شده است. در این فصل بدون کمک گیری از Framework های متداول نسبت به معرفی یک مثال مقدماتی در این زمینه اقدام خواهد شد و سپس در طی فصل آتی، یکی از معروفترین Framework های M-V-VM را بررسی خواهیم کرد.



شکل ۱– نمایی از اجزای کلی برنامه‌ای که با استفاده از الگوی M-V-VM توسعه یافته است.

ساختار پوشه‌های یک برنامه‌ی MVVM

عموما ساختار پوشه‌های یک برنامه‌ی MVVM مطابق شکل بعد است که حداقل از سه پوشه‌ی Models ، ViewModels و Views تشکیل شده است. هر چند هیچ الزامی هم جهت پیروی از این الگوی پوشه‌ها وجود ندارد.



شکل ۲– ساختار پوشه‌های برنامه به همراه نمایی از برنامه در حال اجرا

معرفی برنامه‌ی فصل

در برنامه‌ی فصل جاری، مدل، کلاس شخص (Person) است. ViewModel آن کلاس PeopleViewModel می‌باشد که کار آن وفق دادن اطلاعات مدل به View است. در آن سه شخص جدید ایجاد شده و سپس به View برنامه که از یک User control تشکیل می‌شود، bind خواهند شد. در این View با کلیک بر روی دکمه‌های + و یا – کار افزودن یا کاهش سن هر شخص انتخابی صورت می‌گیرد و همچنین نتیجه‌ی عملیات نیز بلافاصله در سه برچسب واقع شده در کنار ListBox به همراه آیتم انتخابی آن، منعکس می‌گردد.

مدل برنامه

مدل برنامه یا همان کلاس ساده‌ی Person که کد آن‌را در ادامه ملاحظه می‌کنید (تعریف شده در فایل Person.cs پوشه Models پروژه)، معرف خواص سن، نام و نام خانوادگی یک شخص می‌باشد. همچنین برای اینکه تغییرات خواص آن بلافاصله در GUI برنامه منعکس شود اینترفیس استاندارد INotifyPropertyChanged را نیز پیاده سازی کرده است.

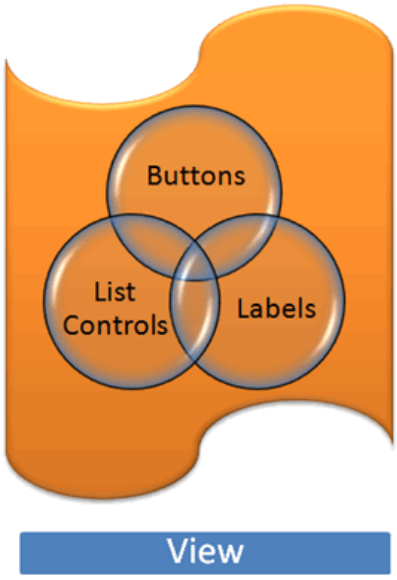
شکل ۵– اجزای تشکیل دهنده‌ی یک Model

Person.cs

```
using System.ComponentModel;

namespace SilverlightApplication48.Models
{
    public class Person : INotifyPropertyChanged
    {
        #region Fields (3)

        private int _age;
        private string _firstName;
        private string _lastName;

        #endregion Fields

        #region Properties (3)

        public int Age
        {
            set
            {
                _age = value;
                if (PropertyChanged == null) return;
                onPropertyChanged("Age");
            }
            get { return _age; }
        }

        public string FirstName
```

```
        {
            set
            {
                _firstName = value;
                if (PropertyChanged == null) return;
                onPropertyChanged("FirstName");
            }
            get { return _firstName; }
        }

        public string LastName
        {
            set
            {
                _lastName = value;
                if (PropertyChanged == null) return;
                onPropertyChanged("LastName");
            }
            get { return _lastName; }
        }

        #endregion Properties

        #region Delegates and Events (1)

        // Events (1)

        public event PropertyChangedEventHandler PropertyChanged;

        #endregion Delegates and Events

        #region Methods (1)

        // Private Methods (1)

        private void onPropertyChanged(string propertyName)
        {
            if (PropertyChanged == null) return;
            PropertyChanged(this,
               new PropertyChangedEventArgs(propertyName));
        }

        #endregion Methods
    }
}
```

View برنامه



شکل ۶– اجزای تشکیل دهنده‌ی یک View

View برنامه با ایجاد یک User control جدید به نام PeopleView.xaml در پوشه Views پروژه معرفی می‌گردد. کدهای Xaml این View ساده به شرح زیر هستند:

PeopleView.xaml

```
<UserControl x:Class="SilverlightApplication48.Views.PeopleView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
      "http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:vm="clr-namespace:SilverlightApplication48.ViewModels"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">

    <UserControl.Resources>
        <vm:PeopleViewModel x:Key="viewModel" />
    </UserControl.Resources>

    <Grid DataContext="{Binding Source={StaticResource viewModel}}">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="150" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <ListBox  ItemsSource="{Binding People}"
                  Grid.Column="0"
```

```
            Margin="5,5,4,5"
            SelectedItem="{Binding SelectedPerson, Mode=TwoWay}">
        <ListBox.ItemTemplate>
            <DataTemplate>
                <StackPanel Orientation="Horizontal">
                    <TextBlock Margin="2"
                               Text="{Binding FirstName}" />
                    <TextBlock Margin="2"
                               Text="{Binding LastName}" />
                    <TextBlock Margin="0 2"
                               Text="[" />
                    <TextBlock Margin="2"
                               Text="{Binding Age}" />
                    <TextBlock Margin="0 2"
                               Text="]" />
                </StackPanel>
            </DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>
    <Grid Grid.Column="1">
        <Grid.RowDefinitions>
            <RowDefinition Height="0.757*" />
            <RowDefinition Height="0.243*" />
        </Grid.RowDefinitions>
        <Grid x:Name="PersonDetails"
              Grid.Row="0"
              DataContext="{Binding SelectedPerson}"
              Margin="5">
            <Grid.RowDefinitions>
                <RowDefinition Height="1*" />
                <RowDefinition Height="1*" />
                <RowDefinition Height="1*" />
            </Grid.RowDefinitions>
            <TextBlock Text="{Binding FirstName}"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center"
                       Grid.Column="0" />
            <TextBlock Text="{Binding LastName}"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center"
                       Grid.Row="1" />
            <TextBlock Text="{Binding Age}"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center"
                       Grid.Row="2" />

        </Grid>
        <StackPanel Orientation="Horizontal"
```

```
                        HorizontalAlignment="Center"
                Grid.Row="1">
            <Button x:Name="button"
                Content="-"
                Width="32"
                Height="32"
                Command="{Binding DecreaseCommand}"
                CommandParameter="{Binding SelectedPerson}"
                >
            </Button>
            <Button x:Name="button1"
                Content="+"
                Width="32"
                Height="32"
                Command="{Binding IncreaseCommand}"
                 CommandParameter="{Binding SelectedPerson}" >
            </Button>
        </StackPanel>
    </Grid>
  </Grid>
</UserControl>
```

همانطور که در کدهای Xaml این View مشاهده می‌کنید هر چند تعدادی از کنترل‌های بصری آن نامگذاری شده‌اند اما اصلا به این امر نیازی نبوده و تمامی تعاملات آن‌ها با اطلاعات برنامه از طریق binding صورت می‌گیرد و در هیچ قسمتی از برنامه (همانند دوران WinForms) ارجاع مستقیمی به این کنترل‌ها وجود نخواهد داشت.

همچنین هیچ روال رخداد گردان کلیکی نیز جهت انتقال رویدادهای برنامه به Code behind این View (که اساسا هیچ Code behind خاصی نیز ندارد) تعریف نشده است و کار انتقال رویدادها به ViewModel برنامه از طریق Commands انجام می‌شود.

این View اطلاعات ViewModel مرتبط را از طریق UserControl.Resources تعریف شده در ابتدای فایل و انتساب آن به DataContext گرید مربوط به User control جاری دریافت می‌کند. پیش از تعریف این منبع نیاز است تا فضای نام SilverlightApplication48.ViewModels که در برگیرنده‌ی کلاس PeopleViewModel است، توسط سطر clr-namespace: SilverlightApplication48.ViewModels اضافه گردد.

با توجه به خواص مشخص شده در binding قسمت‌های مختلف View جاری، طراحی ViewModel ما شکل خواهد گرفت که در ادامه آن‌را بررسی خواهیم کرد.

برنامه ViewModel

ViewModel برنامه با افزودن فایل PeopleViewModel.cs به پوشه ViewModels معرفی می‌گردد. کدهای این کلاس را در ادامه ملاحظه خواهید کرد.

در کلاس PeopleViewModel خواص عمومی ارائه شده، مستقیما در اختیار View قرار خواهند گرفت. برای مثال خاصیت عمومی People که از یک ObservableCollection از کلاس Person است، تشکیل می‌شود. ViewModel هیچ ارجاعی را از View در خود ندارد؛ اما تنها ارجاعی از Model را در خود نگهداری می‌کند.

در ابتدای ایجاد وهله‌ای از کلاس PeopleViewModel ، سه شخص به این مجموعه اضافه می‌شوند. در اینجا برای اینکه از امکانات binding موجود در Silverlight استفاده کنیم، بجای استفاده از یک Generic List متداول، از یک ObservableCollection استفاده گردیده است. همین امکانات binding هستند که الگوی MVVM را تبدیل به انتخاب مناسبی برای کار با Silverlight کرده‌اند و تولید یک برنامه با حداقل گره خوردگی اجزای مختلف آن‌را میسر می‌کنند.

شکل ۷– اجزای تشکیل دهنده‌ی یک ViewModel

PeopleViewModel.cs

```
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Windows.Input;
using SilverlightApplication48.Helper;
using SilverlightApplication48.Models;
```

```
namespace SilverlightApplication48.ViewModels
{
    public class PeopleViewModel : INotifyPropertyChanged
    {
        #region Fields (2)

        private Person _currentPerson;
        private readonly ObservableCollection<Person> _people;

        #endregion Fields

        #region Constructors (1)

        public PeopleViewModel()
        {
            SelectedPerson = new Person();

            _people = new ObservableCollection<Person>
                        {
            new Person { Age = 53, FirstName = "P1", LastName = "L1" },
            new Person { Age = 30, FirstName = "P2", LastName = "L2" },
            new Person { Age = 26, FirstName = "P3", LastName = "L3" },
            };

            DecreaseCommand = new DelegateCommand<Person>(
                                    decrease, canDecrease);
            IncreaseCommand = new DelegateCommand<Person>(
                                    increase, canIncrease);
        }

        #endregion Constructors

        #region Properties (4)

        public ICommand DecreaseCommand { set; get; }

        public ICommand IncreaseCommand { set; get; }

        public ObservableCollection<Person> People
        {
            get { return _people; }
        }

        public Person SelectedPerson
        {
            set
            {
                _currentPerson = value;
                if (PropertyChanged == null) return;
                onPropertyChanged("SelectedPerson");
```

```
            }
            get { return _currentPerson; }
        }

        #endregion Properties

        #region Delegates and Events (1)

        // Events (1)
        public event PropertyChangedEventHandler PropertyChanged;
        #endregion Delegates and Events
        #region Methods (5)
        // Private Methods (5)

        static bool canDecrease(Person person)
        {
            return person != null && person.Age > 0;
        }

        static bool canIncrease(Person person)
        {
            return person != null && person.Age < 60;
        }

        static void decrease(Person person)
        {
            if (person == null) return;
            person.Age--;
        }

        static void increase(Person person)
        {
            if (person == null) return;
            person.Age++;
        }

        private void onPropertyChanged(string propertyName)
        {
            if (PropertyChanged == null) return;
            PropertyChanged(this,
                    new PropertyChangedEventArgs(propertyName));
        }

        #endregion Methods
    }
}
```

عضو جاری انتخاب شده‌ی ListBox برنامه به صورت دو طرفه به خاصیت SelectedPerson بایند می‌شود. به همین جهت نیاز است تا کلاس جاری اینترفیس INotifyPropertyChanged را نیز پیاده سازی نماید.

در Silverlight ، Commands امکان مطلع سازی داده‌های برنامه را از تغییرات صورت گرفته در UI ، میسر می‌سازند. مهم‌ترین نکته‌ای که در طراحی Commands در Silverlight به آن دقت شده است حداقل گره خوردگی میان منبع صادر کننده فرمان و محلی است که فرمان را مدیریت و پردازش می‌کند. به همین جهت در این مثال توانستیم بدون کد نویسی در Code behind صفحه View ، مدیریت رویدادهای رسیده را در کلاس ViewModel انجام دهیم.

دو Command در کلاس PeopleViewModel جهت پاسخ دهی به رویدادهای رسیده کاهش و یا افزایش عدد سن تعریف شده‌اند. نحوه‌ی ایجاد یک وهله‌ی جدید از هر کدام بر اساس کلاس DelegateCommand معرفی شده در فصل آشنایی با سیستم Binding در Silverlight است (که در این مثال در پوشه‌های Helper تعریف شده است).