عنوان: **MVVM و رویدادگردانی**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۲۳ ۱۹:۰۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: MVVM

در دو قسمت قبل به اینجا رسیدیم که بجای شروع به کدنویسی مستقیم در code behind یک View (یک پنجره، یک user control ...)، کلاس مجزای دیگری را به نام ViewModel به برنامه اضافه خواهیم کرد و این کلاس از وجود هیچ فرمی در برنامه مطلع نیست.
بنابراین جهت انتقال رخدادها به ViewModel، بجای روش متداول تعریف روال‌های رخدادگردان در Code behind:

```
<Button  Click="btnClick_Event">Last</Button>
```

آن‌ها را با Commands به ViewModel ارسال خواهیم کرد:

```
<Button Command="{Binding GoLast}">Last</Button>
```

همچنین بجای اینکه مستقیما بخواهیم از طریق نام یک شیء به مثلا خاصیت متنی آن دسترسی پیدا کنیم:

```
<TextBox Name="txtName" />
```

از طریق Binding، اطلاعات مثلا متنی آن‌را به ViewModel منتقل خواهیم کرد:

```
<TextBox Text="{Binding Name}" />
```

و همینجا است که 99 درصد آموزش‌های MVVM موجود در وب به پایان می‌رسند؛ البته پس از مشاهده 10 تا 20 ویدیو و خواندن بیشتر از 30 تا مقاله! و اینجا است که خواهید گفت: فقط همین؟! با این‌ها میشه یک برنامه رو مدیریت کرد؟!
البته همین‌ها برای مدیریت قسمت عمده‌ای از اکثر برنامه‌ها کفایت می‌کنند؛ اما خیلی از ریزه کاری‌ها وجود دارند که به این سادگی‌ها قابل حل نیستند و در طی چند مقاله به آن‌ها خواهیم پرداخت.

سؤال: در همین مثال فوق، اگر متن ورودی در TextBox تغییر کرد، چگونه می‌توان بلافاصله از تغییرات آن در ViewModel مطلع شد؟ قدیم‌ترها می‌شد نوشت:

```
<TextBox TextChanged="TextBox_TextChanged" />
```

اما الان که قرار نیست در code behind کد بنویسیم (تا حد امکان البته)، باید چکار کرد؟
پاسخ: امکان Binding به TextChanged وجود ندارد، پس آن‌را فراموش می‌کنیم. اما همان Binding معمولی را به این صورت هم می‌شود نوشت (همان مثال قسمت قبل):

```
<TextBox Text="{Binding
                            MainPageModelData.Name,
                            Mode=TwoWay,
                            UpdateSourceTrigger=PropertyChanged}" />
```

و نکته مهم آن UpdateSourceTrigger است. اگر روی حالت پیش فرض باشد، ViewModel پس از تغییر focus از این TextBox به کنترلی دیگر، از تغییرات آگاه خواهد شد. اگر آن‌را صریحا ذکر کرده و مساوی PropertyChanged قرار دهیم (این مورد در سیلورلایت 5 جدید است؛ هر چند از روز نخست WPF وجود داشته است)، با هر تغییری در محتوای TextBox، خاصیت MainPageModelData.Name به روز رسانی خواهد شد.
اگر هم بخواهیم این تغییرات آنی‌را در ViewModel تحت نظر قرار دهیم، می‌توان نوشت:

```
using System.ComponentModel;

namespace SL5Tests
{
    public class MainPageViewModel
    {
        public MainPageModel MainPageModelData { set; get; }
        public MainPageViewModel()
        {
            MainPageModelData = new MainPageModel();
            MainPageModelData.Name = "Test1";
            MainPageModelData.PropertyChanged += MainPageModelDataPropertyChanged;
        }

        void MainPageModelDataPropertyChanged(object sender, PropertyChangedEventArgs e)
        {
            switch (e.PropertyName)
            {
                case "Name":
                    //do something
                    break;
            }
        }
    }
}
```

تعریف MainPageModel را در قسمت قبل مشاهده کرده‌اید و این کلاس اینترفیس INotifyPropertyChanged را پیاده سازی می‌کند. بنابراین می‌توان از رویدادگردان PropertyChanged آن در ViewModel هم استفاده کرد.
به این ترتیب همان کار رودیدادگردان TextChanged را اینطرف هم می‌توان شبیه سازی کرد و تفاوتی نمی‌کند. البته با این تفاوت که در ViewModel فقط به اطلاعات به روز موجود در MainPageModelData.Name دسترسی داریم، اما نمی‌دانیم و نمی‌خواهیم هم بدانیم که منبع آن دقیقا کدام شیء رابط کاربری برنامه است.

سؤال: ما قبلا مثلا می‌توانستیم بررسی کنیم که اگر کاربر حین تایپ در یک TextBox بر روی دکمه‌ی Enter کلیک کرد، آن‌گاه برای نمونه، جستجویی بر اساس اطلاعات وارد شده صورت گیرد. الان این فشرده شدن دکمه‌ی Enter را چگونه دریافت و چگونه به ViewModel ارسال کنیم؟
این مورد کمی پیشرفته‌تر از حالت‌های قبلی است. برای حل این مساله ابتدا باید UpdateSourceTrigger یاد شده را مساوی Explicit قرار داد. یعنی اینبار می‌خواهیم نحوه ی به روز رسانی خاصیت MainPageModelData.Name را از طریق Binding خودمان مدیریت کنیم. این مدیریت کردن هم با استفاده از امکاناتی به نام Attached properties قابل انجام است که به آن‌ها Behaviors هم می‌گویند. مثلا:

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
```

```
namespace SL5Tests
{
    public static class InputBindingsManager
    {
        public static readonly DependencyProperty UpdatePropertySourceWhenEnterPressedProperty
                    = DependencyProperty.RegisterAttached(
                                "UpdatePropertySourceWhenEnterPressed",
                                typeof(bool),
                                typeof(InputBindingsManager),
                                new PropertyMetadata(false,
OnUpdatePropertySourceWhenEnterPressedPropertyChanged));

        static InputBindingsManager()
        { }

        public static void SetUpdatePropertySourceWhenEnterPressed(DependencyObject dp, bool value)
        {
            dp.SetValue(UpdatePropertySourceWhenEnterPressedProperty, value);
        }

        public static bool GetUpdatePropertySourceWhenEnterPressed(DependencyObject dp)
        {
            return (bool)dp.GetValue(UpdatePropertySourceWhenEnterPressedProperty);
        }

        private static void OnUpdatePropertySourceWhenEnterPressedPropertyChanged(DependencyObject dp,
    DependencyPropertyChangedEventArgs e)
        {
            var txt = dp as TextBox;
            if (txt == null)
                return;

            if ((bool)e.NewValue)
            {
                txt.KeyDown += HandlePreviewKeyDown;
            }
            else
            {
                txt.KeyDown -= HandlePreviewKeyDown;
            }
        }

        static void HandlePreviewKeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key != Key.Enter) return;

            var txt = sender as TextBox;
            if (txt == null)
                return;

            var binding = txt.GetBindingExpression(TextBox.TextProperty);
            if (binding == null) return;
            binding.UpdateSource();
        }
    }
}
```

تعریف Attached properties یک قالب استاندارد دارد که آن را در کد فوق ملاحظه می‌کنید. یک تعریف به صورت static و سپس تعریف متدهای Get و Set آن. با تغییر مقدار آن که اینجا از نوع bool تعریف شده، متد OnUpdatePropertySourceWhenEnterPressedPropertyChanged به صورت خودکار فراخوانی می‌شود. اینجا است که ما از طریق آرگومان dp به textBox جاری دسترسی کاملی پیدا می‌کنیم. مثلا در اینجا بررسی شده که آیا کلید فشرده شده enter است یا خیر. اگر بله، یک سری فرامین را انجام بده. به عبارتی ما توانستیم، قطعه کدی را به درون شیءایی موجود تزریق کنیم. Txt تعریف شده در اینجا، واقعا همان کنترل TextBox ایی است که به آن متصل شده‌ایم.

و برای استفاده از آن خواهیم داشت:

```
<UserControl x:Class="SL5Tests.MainPage"
    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:SL5Tests"
    mc:Ignorable="d" Language="fa"
    d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Resources>
        <VM:MainPageViewModel x:Name="vmMainPageViewModel" />
    </UserControl.Resources>
    <Grid DataContext="{Binding Source={StaticResource vmMainPageViewModel}}"
          x:Name="LayoutRoot"
          Background="White">
        <TextBox Text="{Binding
                            MainPageModelData.Name,
                            Mode=TwoWay,
                            UpdateSourceTrigger=Explicit}"
                 VerticalAlignment="Top"
                 VM:InputBindingsManager.UpdatePropertySourceWhenEnterPressed="True"  />
    </Grid>
</UserControl>
```

همانطور که مشاهده می‌کنید، UpdateSourceTrigger به Explicit تنظیم شده و سپس InputBindingsManager.UpdatePropertySourceWhenEnterPressed به این کنترل متصل گردیده است. یعنی تنها زمانیکه در متد HandlePreviewKeyDown ذکر شده، متد UpdateSource فراخوانی گردد، خاصیت MainPageModelData.Name به روز رسانی خواهد شد (کنترل آن‌را خودمان در دست گرفته‌ایم نه حالت‌های از پیش تعریف شده).

این روش، روش متداولی است برای تبدیل اکثر حالاتی که Binding و Commanding متداول در مورد آن‌ها وجود ندارد. مثلا نیاز است focus را به آخرین سطر یک ListView از داخل ViewModel انتقال داد. در حالت متداول چنین امری میسر نیست، اما با تعریف یک Attached properties می‌توان به امکانات شیء ListView مورد نظر دسترسی یافت (به آن متصل شد، یا نوعی تزریق)، آخرین عنصر آن‌را یافته و سپس focus را به آن منتقل کرد یا به هر اندیسی مشخص که بعدا در ViewModel به این Behavior از طریق Binding ارسال خواهد شد.