عنوان: **MVVM و رویدادگردانی - قسمت دوم**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۰/۰۷ ۱۰:۲۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: MVVM

قسمت اول این بحث و همچنین پیشنیاز آن‌را در اینجا و اینجا می‌توانید مطالعه نمائید.
همه‌ی این‌ها بسیار هم نیکو! اما ... آیا واقعا باید به ازای هر روال رویدادگردانی یک Attached property نوشت تا بتوان از آن در الگوی MVVM استفاده کرد؟ برای یکی دو مورد شاید اهمیتی نداشته باشد؛ اما کم کم با بزرگتر شدن برنامه نوشتن این Attached properties تبدیل به یک کار طاقت فرسا می‌شود و اشخاص را از الگوی MVVM فراری خواهد داد.
برای حل این مساله، تیم Expression Blend راه حلی را ارائه داده‌اند به نام Interaction.Triggers که در ادامه به توضیح آن پرداخته خواهد شد.
ابتدا نیاز خواهید داشت تا SDK مرتبط با Expression Blend را دریافت کنید: (^)
سپس با فایل System.Windows.Interactivity.dll موجود در آن کار خواهیم داشت.

یک مثال عملی:
فرض کنید می‌خواهیم رویداد Loaded یک View را در ViewModel دریافت کنیم. زمان وهله سازی یک ViewModel با زمان وهله سازی View یکی است، اما بسته به تعداد عناصر رابط کاربری قرار گرفته در View ، زمان بارگذاری نهایی آن ممکن است متفاوت باشد به همین جهت رویداد Loaded برای آن درنظر گرفته شده است. خوب، ما الان در ViewModel نیاز داریم بدانیم که چه زمانی کار بارگذاری یک View به پایان رسیده.
یک راه حل آن‌را در قسمت قبل مشاهده کردید؛ باید برای این کار یک Attached property جدید نوشت چون نمی‌توان Command ایی را به رویداد Loaded انتساب داد یا Bind کرد. اما به کمک امکانات تعریف شده در System.Windows.Interactivity.dll به سادگی می‌توان این رویداد را به یک Command استاندارد ترجمه کرد:

```
<Window x:Class="WpfEventTriggerSample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
        xmlns:vm="clr-namespace:WpfEventTriggerSample.ViewModels"
        Title="MainWindow" Height="350" Width="525">
    <Window.Resources>
        <vm:MainWindowViewModel x:Key="vmMainWindowViewModel" />
    </Window.Resources>
    <Grid DataContext="{Binding Source={StaticResource vmMainWindowViewModel}}">
        <i:Interaction.Triggers>
            <i:EventTrigger EventName="Loaded">
                <i:InvokeCommandAction Command="{Binding DoLoadCommand}"
                                       CommandParameter="I am loaded!" />
            </i:EventTrigger>
        </i:Interaction.Triggers>

        <TextBlock Text="Testing InvokeCommandAction..."
                   Margin="5" VerticalAlignment="Top" />
    </Grid>
</Window>
```

ابتدا ارجاعی به اسمبلی System.Windows.Interactivity.dll باید به پروژه اضافه شود. سپس فضای نام xmlns:i باید به فایل XAML جاری مطابق کدهای فوق اضافه گردد. در نهایت به کمک Interaction.Triggers آن، ابتدا نام رویداد مورد نظر را مشخص می‌کنیم (EventName) و سپس به کمک InvokeCommandAction، این رویداد به یک Command استاندارد ترجمه می‌شود.
ViewModel این View هم می‌تواند به شکل زیر باشد که با کلاس DelegateCommand آن در پیشنیازهای بحث جاری آشنا شده‌اید.

```
using WpfEventTriggerSample.Helper;
```

```
namespace WpfEventTriggerSample.ViewModels
{
    public class MainWindowViewModel
    {
        public DelegateCommand<string> DoLoadCommand { set; get; }
        public MainWindowViewModel()
        {
            DoLoadCommand = new DelegateCommand<string>(doLoadCommand, canDoLoadCommand);
        }

        private void doLoadCommand(string param)
        {
            //do something
        }

        private bool canDoLoadCommand(string param)
        {
            return true;
        }
    }
}
```

به این ترتیب حجم قابل ملاحظه‌ای از کد نویسی Attached properties مورد نیاز، به ساده‌ترین شکل ممکن، کاهش خواهد یافت. بدیهی است این Interaction.Triggers را جهت تمام عناصر UI ایی که حداقل یک رویداد منتسب تعریف شده داشته باشند، می‌توان بکار گرفت؛ مثلا تبدیل رویداد Click یک دکمه به یک Command استاندارد:

```
<Button>
            <i:Interaction.Triggers>
                <i:EventTrigger EventName="Click">
                    <i:InvokeCommandAction Command="{Binding DoClick}"
                                           CommandParameter="I am loaded!" />
                </i:EventTrigger>
            </i:Interaction.Triggers>
</Button>
```

نظرات خوانندگان

نویسنده: محمد صاحب
تاریخ: ۱۳۹۰/۱۰/۰۷ ۱۱:۱۱:۱۰

ممنون

سوال اول:
این قسمت رو من درست متوجه نشدم
<>
رویداد کلیک رو که میشه مستقیم بایند کرد؟

سوال دوم:
فکر میکنید موارد ی که کار با این الگو رو راحت میکنن بصورت توکار برای WPF و SL اضافه بشه.
دقیقا چیزی که تو MVC داریم مثلا ساختار پروژه و نحوه نامگذاری (اضافه کردن controller به نام و...)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۰/۰۷ ۱۱:۳۴:۰۶

- برای مثال نمی‌شود نوشت Click = Binding DoLoadCommand ، به همین جهت نیاز هست تا این event handler را تبدیل به یک command استاندارد کرد تا در ViewModel قابل دسترسی شود.
در کل هدف من یک مثال کلی بود که بگم این همه جا کاربرد دارد، مثلا اینطوری هم میشه با آن کار کرد.
- مایکروسافت همین الان یک فریم ورک MVVM تمام عیار به نام PRISM دارد: (^)