عنوان: مروری سریع بر اصول مقدماتی MVVM
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۲۱ ۰۰:۵۰:۰۰
آدرس: www.dotnettips.info
گروه‌ها: MVVM

در قسمت قبل
، فلسفه وجودی MVVM و MVC و امثال آن‌را به بیانی ساده مطالعه کردید. همچنین به اینجا رسیدیم که بجای نوشتن روال رخدادگردان، از Commands استفاده کنید.
در این قسمت «تفکر MVVM ایی» بررسی خواهد شد! بنابراین سطح این قسمت را هم مقدماتی درنظر بگیرید.

در سیستم متداول مایکروسافتی ما همیشه یک فرم داریم به همراه یک سری کنترل. برای استفاده از این‌ها هم در فایل code behind فرم مرتبط، امکان دسترسی به این کنترل‌ها وجود دارد. مثلا textBox1.Text یعنی ارجاعی مستقیم به شیء textBox1 و سپس دسترسی به خاصیت متنی آن.
«تفکر MVVM ایی» می‌گه که: خیر! اینکار رو نکنید؛ ارجاع مستقیم به یک کنترل روش کار من نیست! فرم رو طراحی کنید؛ برای هیچکدام از کنترل‌ها هم نامی را مشخص نکنید (برخلاف رویه متداول). یک فایل درست کنید به نام Model ، داخل آن معادل textBox1.Text را که می‌خواهید استفاده کنید، پیش بینی و تعریف کنید؛ مثلا Public string Name . همین!
ما نمی‌خواهیم بدانیم که اصلا textBox1 وجود خارجی دارد یا نه. ما فقط با خاصیت متنی آن که در ادامه نحوه‌ی سیم کشی آن‌را هم بررسی خواهیم کرد، کار داریم.
بنابراین بجای اینکه بنویسید:

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

که ممکن است بعدا وسوسه شوید تا از txtName.Text آن استفاده کنید، بنویسید:

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

این مهم‌ترین قسمت «تفکر MVVM ایی» به زبان ساده است. یعنی قرار است تا حد ممکن از Binding استفاده کنیم. مثلا در قسمت قبل هم دیدید که بجای نوشتن روال رخدادگردان، فرمان مرتبط با آن‌را به جای دیگری Bind کردیم.

بنابراین تا اینجا Model ما به این شکل خواهد بود:

```
using System.ComponentModel;

namespace SL5Tests
{
    public class MainPageModel : INotifyPropertyChanged
    {
        string _name;
        public string Name
        {
            get { return _name; }
            set
            {
                if (_name == value) return;
                _name = value;
                raisePropertyChanged("Name");
            }
        }
```

```
        public event PropertyChangedEventHandler PropertyChanged;
        void raisePropertyChanged(string propertyName)
        {
            var handler = PropertyChanged;
            if (handler == null) return;
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

سؤال مهم:

تا اینجا یک فایل Model داریم که خاصیت Name در آن تعریف شده؛ یک فرم (View) هم داریم که فقط در آن نوشته شده Binding Name. الان این‌ها چطور به هم متصل خواهند شد؟

پاسخ: اینجا است که کلاس دیگری به نام ViewModel (همان فایل Code behind قدیمی است با این تفاوت که به هیچ فرم خاصی گره نخورده است و اصلا نمی‌داند که در برنامه فرمی وجود دارد یا نه)، کار خودش را شروع خواهد کرد:

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

ما در این کلاس یک وهله از MainPageModel را ایجاد خواهیم کرد. اگر فرمی (که ما دقیقا نمی‌دانیم کدام فرم) در برنامه نیاز به یک ViewModel بر اساس مدل یاد شده داشت، می‌تواند آن‌را مورد استفاده قرار دهد. مقدار دهی آن در ViewModel موجب مقدار دهی خاصیت Text در فرم مرتبط خواهد شد و برعکس (البته به شرطی که مدل ما INotifyPropertyChanged را پیاده سازی کرده باشد و در فرم برنامه Binding Mode دو طرفه تعریف شود).

در قسمت بعد هم کار اتصال نهایی صورت می‌گیرد:

ابتدا xmlns:VM تعریف می‌شود تا بتوان به ViewModelها در طرف XAML دسترسی پیدا کرد. سپس در قسمت مثلا UserControl.Resources، این ViewModel را تعریف کرده و به عنوان DataContext بالاترین شیء فرم مقدار دهی خواهیم کرد:

```
<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=PropertyChanged}" />
    </Grid>
</UserControl>
```

اکنون اگر یک breakpoint روی این سطر Binding قرار دهیم و برنامه را اجرا کنیم، جزئیات این سیم کشی را در عمل بهتر می‌توان مشاهده کرد:

```
12    <Grid DataContext="{Binding Source={StaticResource vmMainPageViewModel}}"
13          x:Name="LayoutRoot"
14          Background="White">
15        <TextBox Text="{Binding
16                              MainPageModelData.Name,
```

100 %

XAML | Design | TextBox UserControl/Grid/TextBox

Locals

Name	Value
BindingState	{UpdatingTarget}
Action	UpdatingTarget
Binding	{System.Windows.Data.Binding}
BindingExpression	{System.Windows.Data.BindingExpression}
[System.Windows.Data.BindingExpre	{System.Windows.Data.BindingExpression}
base	{System.Windows.Data.BindingExpression}
DataItem	{SL5Tests.MainPageViewModel}
MainPageModelData	{SL5Tests.MainPageModel}
_name	"Test1"
Name	"Test1"
PropertyChanged	{System.ComponentModel.PropertyChangedEventHandler}

البته این قابلیت قرار دادن breakpoint روی Bindingهای تعریف شده در View فعلا به سیلورلایت 5 اضافه شده و هنوز در WPF موجود نیست.

حداقل مزیتی را که اینجا می‌توان مشاهده کرد این است که فایل MainPageViewModel چون نمی‌داند که قرار است در کدام View وهله سازی شود، به سادگی در Viewهای دیگر نیز قابل استفاده خواهد بود یا تغییر و تعویض کلی View آن کار ساده‌ای است.
Commanding قسمت قبل را هم اینجا می‌شود اضافه کرد. تعاریف DelegateCommandهای مورد نیاز در ViewModel قرار می‌گیرند. مابقی عملیات تفاوتی نمی‌کند و یکسان است.

نظرات خوانندگان

نویسنده: hossein moradinia
تاریخ: ۱۳۹۰/۰۹/۲۱ ۱۲:۲۰:۱۳

کاملا واضحه که الگوی MVVM برای جداسازی رابط کاربری نرم افزار (View) از مدل برنامه طراحی شده. هچنین میدونیم که الگویی به نام Repository وجود داره که بر روی ORM برای مثال Entity Framework پیاده میشه و عملکرد این دو الگو متفاوت هست.مزایای استفاده از Repository هم که مشخصه...
حال سوال اینجاست که آیا میشه از این دو الگو در کنار هم دیگر استفاده کرد؟!!!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۲۱ ۱۲:۳۸:۰۲

بله. در همان ViewModel عنوان شده، الگوی مخزن را با توجه به وجود مثلا شیء MainPageModelData فراخوانی و مقدار دهی کنید.

نویسنده: Ahmadxml
تاریخ: ۱۳۹۰/۰۹/۲۱ ۱۳:۲۸:۲۲

بسیار عالی و قابل فهم

نویسنده: alireza
تاریخ: ۱۳۹۰/۰۹/۲۱ ۱۹:۱۸:۴۷

سلام
تشکر از مطالب بسیار مفیدتون
من چندین ساله دارم برنامه نویسی میکنم و جدیدن با تکنولوژی WPF آشناشدم. متاسفانه هر مرجعی که برای یادگیری این اصول (MVVM , WPF,...) که بکار بردم در اول کار مطالب خیلی پیش پا افتاده رو بیان میکنند و بدون گفتن پیش زمینه های لازم وارد مباحث بسیار سنگین میشن. که باعث میشه آموزنده از مطلب زده بشه.(یه جورایی هم احساس حقارت در مورد سواد کم خودش بهش دست بده) در هر صورت من علاقه بسیار زیادی به برنامه نویسی داشتم و دارم وخیلی دوست دارم با تکنولوژی های جدید بیشتر کار کنم و سبک کاریم رو بروز کنم. از شما که در این زمینه تجربه کافی دارین میخوام لطف کنین یک منبع و مرجع برای یادگیری این مباحث (مباحث جدید که یادگیریش واسه برنامه نویسی الان از نون شب واجب تره) چه فارسی چه انگلیسی معرفی کنین. لازم به ذکره که مطالب آموزشی که خود شما میذارید تقریبا از سواد الان من فراتره و خیلی از قسمت هاشو درک نمیکنم(که قطعا به خاطر سواد کم من در این زمینه است).
پیشاپیش از لطفتون ممنونم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۲۱ ۱۹:۳۰:۱۳

پیشنیاز MVVM مباحث Binding در Silverlight و WPF است. یک کتاب فارسی رو در این زمینه در اینجا می‌تونید دریافت کنید: (^)
مرتبط با سیلورلایت است اما ... مباحث کلی آن با WPF تفاوتی ندارد و اصول یکی است.

نویسنده: Alimomen54
تاریخ: ۱۳۹۰/۰۹/۲۶ ۱۳:۵۹:۳۵

سلام
از مطالب خوبتون تشکر می کنم. دیگه تقریبا مشتری ثابت و ساعتی سایتتون شدم.
wpf & mvvm هنوز تبدیل به یه ابزار کامل برای تولید یه برنامه کاربردی حرفه ای نشده. من پوستم کنده شد تا تونستم یه برنامه

کامل باهاش بنویسم. واسه راتباطش با ssrs چه مشقاتی که نکشیدم.
ناچار شدم یه فرم ویندوزی به برنامه اضافه کنم و گزارش را توی اون نمایش بدم.
راهم درسته فعلا؟ در نسخه 2011 فکری به حال این مشکل نشده؟
راستی چطور میشه توسط یه کلید و بدون نوشتم کد یه ویو جدید را نمایش داد. راهی برای بایند کردن کلید به ویو وجود داره.
بدون نوشتن Command ?

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۲۶ ۱۷:۱۴:۲۴

احتمالا این مطلب راهبری برای شما مفید باشد.

نویسنده: saman
تاریخ: ۱۳۹۱/۰۴/۰۴ ۲۳:۵۵

سلام.با تشکر از مطالب مفیدتون.راستش رو بخواین من نمیدونم باید اینجا سوالم رو مطرح کنم یانه؟چون تاپیک مرتبطتری پیدا نکردم.
من با الگوی MVVMکار میکنم.برای نمایش خطاهای اعتبار سنجی هم از IDataErrorInfo استفاده کردم.
مشکل من اینجاست که وقتی یک پروپرتی از نوع int رو به یکی از تکست باکسهام بایند میکنم و میخوام که کاربر مقدار اون فیلد رو همیشه پر کنه یعنی not nullable
هستش.وقتی متن داخل تکست باکس رو پاک میکنم بجای خطای در نظر گرفته شده براش عبارت زیر داخل tooltipنمایش داده میشه:
value "" could not be converted
ممنون میشم اگه راهنماییم کنین.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۵ ۰:۱۳

باید از ValidationRules استفاده کنید. مثلا : (^)