عنوان: آشنایی با Catel MVVM Frameowork
نویسنده: محمّد زارع
تاریخ: ۱۳۹۲/۰۳/۱۳ ۱۶:۱۵
آدرس: www.dotnettips.info
گروه‌ها: MVVM, WPF, Catel

در این مقاله به بررسی اولیه فریمورک Catel و برخی ویژگی‌های آن خواهیم پرداخت.
همانطور که می‌دانید فریمورک‌های متعددی برای MVVM به وجود آمده اند، مانند MVVM Light یا Caliburn و Chinch و ... که هر کدام از آن‌ها دارای ویژگی هایی می‌باشند اما Catel تنها یک فریمورک برای MVVM نیست بلکه دارای قسمت‌های دیگری مانند کنترل‌های اختصاصی و سرویس‌های متعدد و پرکاربرد و Extensionهای مفید و ... نیز می‌باشد که کار توسعه یک برنامه MVVM را فوق العاده لذتبخش می‌کند.
برای شروع کار با این فریمورک ابتدا بایستی قالب پروژه را از این آدرس دریافت نمایید . بعد از دریافت و نصب آن یک زیرگروه جدید به نام Catel به قسمت افزودن پروژه جدید اضافه خواهد شد که شامل قالب پروژه برای WPF و Silverlight و Windows Phone و Windows Store می باشد. در این قسمت گزینه WPF Application with Catel را انتخاب نمایید و پروژه را ایجاد کنید.
بعد از ایجاد پروژه نوبت به نصب بسته های nuget مورد نیاز Catel می رسد. تنها بسته مورد نیاز Catel.Extensions.Controls می باشد که به صورت خودکار بسته های Catel.MVVM و Catel.Core را نیز نصب خواهد کرد. البته بسته‌های دیگری مانند Catel.Extensions.Prism, Catel.Extensions.FluentValidation و Catel.Extensions.Data و Catel.Fody و ... نیز برای این فریمورک وجود دارد که در این مطلب به آن‌ها نیازی نداریم.
اکنون ساختار اصلی پروژه ما ایجاد شده است. در این ساختار پوشه‌های Views، Models و ViewModels به صورت پیش فرض وجود دارند. Catel برای برقراری ارتباط بین View و ViewModel از IViewModelLocator ، IViewLocator و یکسری قواعد نام گذاری پیروی میکند تا نیاز به رجیستر کردن تک تک ویوها و ویومدل‌ها به صورت دستی نباشد که البته این قواعد قابل تغییر و شخصی سازی هستند. قرارداد پیش فرض برای پروژه‌های کوچک ممکن است مناسب باشد ولی در پروژه‌های بزرگ نیاز به سفارشی سازی دارد که در قسمت‌های بعد به آن خواهیم پرداخت.

View و ViewModel:

برای ایجاد یک ViewModel جدید، باید از منوی Add New Item قسمت Catel گزینه ViewModel (Catel) را انتخاب نمایید. با توجه به code snippet های تهیه شده برای این فریمورک، کار تهیه ViewModelها فوق العاده سریع انجام می‌شود. به عنوان مثال برای اضافه کردن یک Command در ویومدل، از vmcommand و یا vmcommandwithcanexecute و برای ایجاد پروپرتی هم از vmprop و vmpropchanged میتوان استفاده نمود. همانطور که ملاحظه می‌کنید نام این snippetها کاملا واضح می‌باشد و نیاز به توضیح اضافی ندارند.
همینطور برای ایجاد یک View گزینه DataWindow (WPF with Catel) را انتخاب نمایید. ViewModelها در Catel از کلاس پایه ViewModelBase و Viewها نیز از کلاس DataWindow مشتق می‌شوند.
DataWindow یک Window پیشرفته با قابلیت هایی مانند افزودن خودکار دکمه‌های Ok / Cancel یا Ok / Cancel / Apply یا Close می‌باشد که می‌تواند باعث تسریع روند ایجاد Windowهای تکراری شود. اما اگر به هیچ کدام از این دکمه‌های ذکر شده نیاز نداشتید DataWindowMode.Custom را انتخاب می‌کنید. نشان دادن Validation در بالای پنجره به صورت popup نیز یکی دیگر از قابلیت‌های این Window پیشرفته است. البته DataWindow دارای overloadهای مختلفی است که می‌توانید به کمک آن ویژگی‌های ذکر شده را فعال یا غیر فعال کنید.
حال برای درک بهتر commandها و نحوه تعریف و بکارگیری آن‌ها یک command جدید در MainWindowViewModel با استفاده از vmcommand ایجاد کنید. مانند قطعه کد زیر:

```
public class MainWindowViewModel : ViewModelBase
    {
        public MainWindowViewModel()
            : base()
        {
            ShowPleaseWait = new Command(OnShowPleaseWaitExecute);
        }

        public override string Title { get { return "View model title"; } }

        public Command ShowPleaseWait { get; private set; }
        private void OnShowPleaseWaitExecute()
        {
            var pleaseWaitService = GetService<IPleaseWaitService>();
```

```
            pleaseWaitService.Show(() =>
            {
                Thread.Sleep(3000);
            });
        }
    }
```

در داخل بدنه این command از PleaseWaitService استفاده کردیم که در ادامه توضیح داده خواهد شد. در MainView نیز یک button اضافه کنید و پروپرتی Command آن را به صورت زیر تنظیم کنید:

```
<Button Margin="6"
                Command="{Binding ShowPleaseWait}"
                Content="Show PleaseWait!" />
```

اکنون با فشردن button کد داخل بدنه command اجرا خواهد شد.

سرویس ها:

کتابخانه Catel.MVVM دارای سرویس‌های مختلف و پرکاربردی می‌باشد که در ادامه به بررسی آن‌ها خواهیم پرداخت:

PleaseWaitService: از این سرویس برای نشان دادن یک loading به کاربر در حین انجام یک کار سنگین استفاده می‌شود و نحوه استفاده از آن به صورت زیر است:

```
var pleaseWaitService = GetService<IPleaseWaitService>();
pleaseWaitService.Show(() =>
{
        Thread.Sleep(3000);
});
```

UIVisualizerService: از این سرویس برای باز کردن پنجره‌های برنامه استفاده می‌شود. هر View در برنامه دارای یک ViewModel می باشد. برای باز کردن View ابتدا یک نمونه از ViewModel مربوطه را ایجاد میکنیم و با دادن viewmodel به متد Show یا ShowDialog پنجره مورد نظر را باز میکنیم.

```
var uiService = GetService<IUIVisualizerService>();
var viewModel = new AnotherWindowViewModel();
uiService.Show(viewModel);
```

OpenFileService: برای نشان دادن OpenFileDialog جهت باز کردن یک فایل در برنامه.

```
var openFileService = GetService<IOpenFileService>();
openFileService.Filter = "ZIP files (*.zip)|*.zip";
openFileService.IsMultiSelect = false;
openFileService.Title = "Open file";
if (openFileService.DetermineFile())
{
       // ?
}
```

SaveFileService: برای نشان دادن SaveFileDialog جهت ذخیره سازی.

```
var saveFileService = GetService<ISaveFileService>();
saveFileService.Filter = "ZIP files (*.zip)|*.zip";
saveFileService.FileName = "test";
saveFileService.Title = "Save file";
if (saveFileService.DetermineFile())
{
       // ?
```

```
}
```

ProcessService: برای اجرا کردن یک process. به عنوان مثال برای باز کردن ماشین حساب ویندوز به صورت زیر عمل می‌کنیم:

```
var processService = GetSetvice<IProcessService>();
processService.StartProcess(@"C:\Windows\System32\calc.exe");
```

SplashScreenService: برای نشان دادن SplashScreen در ابتدای برنامه هایی که سرعت بالا آمدن پایینی دارند.

```
var splashScreenService = GetService<ISplashScreenService>();
splashScreenService.Enqueue(new ActionTask("Creating the shell", OnCreateShell));
splashScreenService.Enqueue(new ActionTask("Initializing modules", OnInitializeModules));
splashScreenService.Enqueue(new ActionTask("Starting application", OnStartApplication));
```

MessageService: برای نشان دادن MessageBox به کاربر.

```
var messageService = GetService<IMessageService>();
if (messageService.Show("Are you sure?", "?", MessageButton.YesNo, MessageImage.Warning) ==
MessageResult.Yes)
{
	// ?
}
```

همانطور که ملاحظه کردید اکثر کارهای مورد نیاز یک پروژه با کمک سرویس‌های ارائه شده در این فریمورک به آسانی انجام می‌شود.

دریافت مثال و پروژه کامل این قسمت: TestApp.zip