عنوان: **یکپارچه سازی اعتبارسنجی EF Code first با امکانات WPF و حذف کدهای تکرای INotifyPropertyChanged**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۸ ۲۲:۴۷
آدرس: www.dotnettips.info
برچسب‌ها: Design patterns, AOP, Dependency Injection, Entity framework, MVVM, WPF

در لابلای توضیحات قسمت‌های قبل، به نحوه استفاده از کلاس‌های پایه‌ای که اعتبارسنجی یکپارچه‌ای را با WPF و EF Code first در قالب پروژه WPF Framework ارائه می‌دهند، اشاره شد. در این قسمت قصد داریم جزئیات بیشتری از پیاده سازی آن‌ها را بررسی کنیم.

بررسی سطح بالای مکانیزم‌های اعتبارسنجی و AOP بکارگرفته شده

در حین کار با قالب پروژه WPF Framework، هنگام طراحی Modelهای خود (تفاوتی نمی‌کند که Domain model باشند یا صرفا Model متناظر با یک View)، نیاز است دو مورد را رعایت کنید:

```
[ImplementPropertyChanged] // AOP
public class LoginPageModel : DataErrorInfoBase
```

الف) کلاس مدل شما باید مزین به ویژگی ImplementPropertyChanged شود.
ب) از کلاس پایه DataErrorInfoBase مشتق گردد

البته اگر به کلاس‌های Domain model برنامه مراجعه کنید، صرفا مشتق شدن از BaseEntity را ملاحظه می‌کنید:

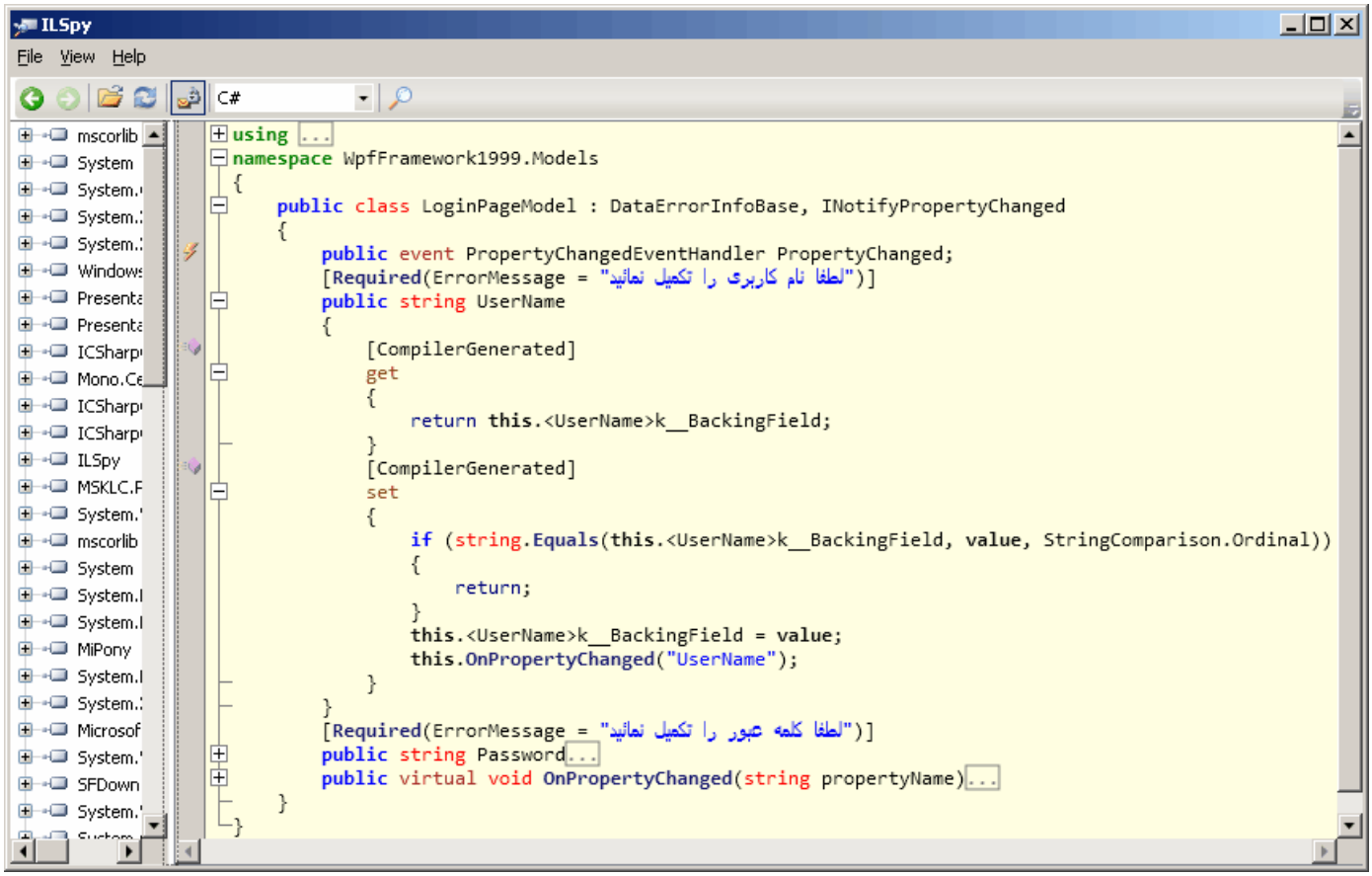
```
public class User : BaseEntity
```

علت این است که دو نکته یاد شده در کلاس پایه BaseEntity پیشتر پیاده سازی شده‌اند:

```
[ImplementPropertyChanged] // AOP
public abstract class BaseEntity : DataErrorInfoBase //پیاده سازی خودکار سیستم اعتبارسنجی یکپارچه
```

بررسی جزئیات مکانیزم AOP بکارگرفته شده

بسیار خوب؛ این‌ها چطور کار می‌کنند؟!
ابتدا نیاز است مطلب « معرفی پروژه NotifyPropertyWeaver » را یکبار مطالعه نمائید. خلاصه‌ای جهت تکرار نکات مهم آن:
ویژگی ImplementPropertyChanged به ابزار Fody اعلام می‌کند که لطفا کدهای تکراری INotifyPropertyChanged را پس از کامپایل اسمبلی جاری، بر اساس تزریق کدهای IL متناظر، به اسمبلی اضافه کن. این روش از لحاظ کارآیی و همچنین تمیز نگه داشتن کدهای نهایی برنامه، فوق العاده است.
برای بررسی کارکرد آن نیاز است اسمبلی مثلا Models را دی‌کامپایل کرد:



همانطور که ملاحظه می‌کنید، کدهای تکراری INotifyPropertyChanged به صورت خودکار به اسمبلی نهایی اضافه شده‌اند.
البته بدیهی است که استفاده از Fody الزامی نیست. اگر علاقمند هستید که این اطلاعات را دستی اضافه کنید، بهتر است از کلاس پایه BaseViewModel قرار گرفته در مسیر MVVM\BaseViewModel.cs پروژه Common استفاده نمائید.
در این کلاس، پیاده سازی‌های NotifyPropertyChanged را بر اساس متدهایی که یک رشته را به عنوان نام خاصیت دریافت می‌کنند و یا متدی که امکان دسترسی strongly typed به نام رشته را میسر ساخته است، ملاحظه می‌کنید.

```
/// <summary>
  /// تغییر مقدار یک خاصیت را اطلاع رسانی خواهد کرد
  /// </summary>
  /// <param name="propertyName">نام خاصیت</param>
  public void NotifyPropertyChanged(string propertyName)

  /// <summary>
  /// تغییر مقدار یک خاصیت را اطلاع رسانی خواهد کرد
  /// </summary>
  /// <param name="expression">نام خاصیت مورد نظر</param>
  public void NotifyPropertyChanged(Expression<Func<object>> expression)
```

برای مثال در اینجا خواهیم داشت:

```
public class AlertConfirmBoxViewModel : BaseViewModel
    {
        AlertConfirmBoxModel _alertConfirmBoxModel;
        public AlertConfirmBoxModel AlertConfirmBoxModel
        {
            set
            {
                _alertConfirmBoxModel = value;
                NotifyPropertyChanged("AlertConfirmBoxModel");
                // ویا ....
                NotifyPropertyChanged(()=>AlertConfirmBoxModel);
            }
            get { return _alertConfirmBoxModel; }
```

```
        }
```

هر دو حالت استفاده از متدهای NotifyPropertyChanged به همراه کلاس پایه BaseViewModel در اینجا ذکر شده‌اند. حالت استفاده از Expression به علت اینکه تحت نظر کامپایلر است، در دراز مدت نگه‌داری برنامه را ساده‌تر خواهد کرد.

بررسی جزئیات اعتبارسنجی‌های تعریف شده

EF دارای یک سری ویژگی مانند Required و امثال آن است. WPF دارای اینترفیسی است به نام IDataErrorInfo. این دو را باید به نحوی به هم مرتبط ساخت که پیاده سازی‌های مرتبط با آن‌ها را در مسیرهای WpfValidation\DataErrorInfoBase.cs و WpfValidation\ValidationHelper.cs پروژه Common می‌توانید ملاحظه نمائید.

```
<TextBox Text="{Binding Path=ChangeProfileData.UserName,
Mode=TwoWay,UpdateSourceTrigger=PropertyChanged,
 NotifyOnValidationError=true, ValidatesOnExceptions=true, ValidatesOnDataErrors=True,
TargetNullValue=''}"  />
```

برای نمونه در اینجا خاصیت Text یک TextBox به خاصیت UserName شیء ChangeProfileData تعریف شده در ViewModel تغییر اطلاعات کاربری برنامه مقید شده است.
همچنین حالت‌های بررسی اعتبارسنجی آن نیز به PropertyChanged تنظیم گردیده است. در این حالت WPF به تعاریف شیء ChangeProfileData مراجعه کرده و برای نمونه اگر این شیء اینترفیس IDataErrorInfo را پیاده سازی کرده بود، نام خاصیت جاری را به آن ارسال و از آن خطاهای اعتبارسنجی متناظر را درخواست می‌کند. در اینجا وقت خواهیم داشت تا بر اساس ویژگی‌ها و Data annotaions اعمالی، کار اعتبارسنجی را انجام داده و نتیجه را بازگشت دهیم.
خلاصه‌ی تمام این اعمال و کلاس‌ها، در کلاس پایه DataErrorInfoBase این قالب پروژه قرار گرفته‌اند. بنابراین تنها کاری که باید صورت گیرد، مشتق کردن کلاس مدل مورد نظر از آن می‌باشد.
همچنین باید دقت داشت که نمایش اطلاعات خطاهای حاصل از اعتبارسنجی در این قالب پروژه بر اساس امکانات قالب متروی MahApps.Metro انجام می‌گیرد (این مورد از Silverlight toolkit به ارث رسیده است) و در حالت کلی خودکار نیست؛ اما در اینجا نیازی به کدنویسی اضافه‌تری ندارد.

به علاوه باید دقت داشت که این مورد ویژه را باید بر اساس آخرین Build کتابخانه MahApps.Metro که به‌روزتر است دریافت و استفاده کرد. در اینجا با پارامتر Pre ذکر شده است.

```
PM> Install-Package MahApps.Metro -Pre
```