عنوان: **INPC استاندارد با بهره گیری از صفت CallerMemberName**
نویسنده: محسن افشین
تاریخ: ۱۳۹۴/۰۴/۱۷ ۱۵:۴۰
آدرس: www.dotnettips.info
گروه‌ها: WPF, .NET 4.5, C# 5.0

یکی از Attributeهای بسیار کاربردی که در سی شارپ 5 اضافه شد CallerMemberNameAttribute بود. این صفت به یک متد اجازه میدهد که از فراخواننده‌ی خود مطلع شود. این صفت را می‌توان بر روی یک پارامتر انتخابی که مقدار پیش‌فرضی دارد اعمال نمود.

استفاده از این صفت هم بسیار ساده است:

```
private void A ( [CallerMemberName] string callerName = "")
{
  Console.WriteLine("Caller is " + callerName);
}

private static void B()
{
        // let's call A
        A();
}
```

در کد فوق، متد A به راحتی می‌تواند بفهمد چه کسی آن را فراخوانی کرده است. از جمله کاربردهای این صفت در ردیابی و خطایابی است.

ولی یک استفاده‌ی بسیار کاربردی از این صفت، در پیاده سازی رابط INotifyPropertyChanged می‌باشد.

معمولا هنگام پیاده سازی INotifyPropertyChanged کدی شبیه به این را می‌نویسیم:

```
public class PersonViewModel : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        private void OnPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
        }

        private string name;
        public string Name
        {
            get { return name; }
            set
            {
                this.name = value;
                OnPropertyChanged("Name");
            }
        }
    }
```

یعنی در Setter معمولا نام ویژگی ای را که تغییر کرده است، به متد OnPropertyChanged می‌فرستیم تا اطلاع رسانی‌های لازم انجام پذیرد. تا اینجای کار همه چیز خوب و آرام است. اما به محضی که کد شما کمی طولانی شود و شما به دلایلی نیاز به Refactor کردن کد و احیانا تغییر نام ویژگی‌ها را پیدا کنید، آن موقع مسائل جدیدی بروز پیدا می‌کند.

برای مثال فرض کنید پس از نوشتن کلاس PersonViewModel تصمیم می‌گیرد نام ویژگی Name را به FirstName تغییر دهید؛ چرا که می‌خواهید اجزای نام یک شخص را به صورت مجزا نگهداری و پردازش کنید. پس احتمالا با زدن کلید F2 روی فیلد name آن را به

firstName و ویژگی Name را به FirstName تغییر نام می‌دهید. همانند کد زیر:

```
private string firstName;
public string FirstName
{
            get { return firstName; }
            set
            {
                this.firstName = value;
                OnPropertyChanged("Name");
            }
}
```

برنامه را کامپایل کرده و در کمال تعجب می‌بینید که بخشی از برنامه درست رفتار نمی‌کند و تغییراتی که در نام کوچک شخص توسط کاربر ایجاد می‌شود به درستی بروزرسانی نمی‌شوند. علت ساده است: ما کد را به صورت اتوماتیک Refactor کرده ایم و گزینه‌ی Include String را در حین Refactor، در حالت پیشفرض غیرفعال رها کرده‌ایم. پس جای تعجبی ندارد که در هر جای کد که رشته‌ای به نام "Name" با ماهیت نام شخص داشته ایم، دست نخورده باقی مانده است. در واقع در کد تغییر یافته، هنگام تغییر FirstName، ما به سیستم گزارش می‌کنیم که ویژگی Name (که اصلا وجود ندارد) تغییر یافته است و این یعنی خطا.

حال احتمال بروز این خطا را در ViewModel هایی با ده‌ها ویژگی و ترکیب‌های مختلف در نظر بگیرید. پس کاملا محتمل است و برای خیلی از دوستان این اتفاق رخ داده است.

و اما راه حل چیست؟ به کارگیری صفت CallerMemberName

بهتر است که یک کلاس انتزاعی برای تمام ViewModelهای خود داشته باشیم و پیاده سازی جدید INPC را در درون آن قرار دهیم تا براحتی VMهای ما از آن مشتق شوند:

```
public abstract class ViewModelBase : INotifyPropertyChanged
{
        public event PropertyChangedEventHandler PropertyChanged;

        protected void OnPropertyChanged([CallerMemberName] string propertyName = "")
        {
            OnPropertyChangedExplicit(propertyName);
        }

        protected void OnPropertyChanged<TProperty>(Expression<Func<TProperty>> projection)
        {
            var memberExpression = (MemberExpression)projection.Body;
            OnPropertyChangedExplicit(memberExpression.Member.Name);
        }

        void OnPropertyChangedExplicit(string propertyName)
        {
            this.CheckPropertyName(propertyName);

            PropertyChangedEventHandler handler = this.PropertyChanged;

            if (handler != null)
            {
                var e = new PropertyChangedEventArgs(propertyName);
                handler(this, e);
            }
        }

        #region Check property name

        [Conditional("DEBUG")]
        [DebuggerStepThrough]
        public void CheckPropertyName(string propertyName)
        {
            if (TypeDescriptor.GetProperties(this)[propertyName] == null)
                throw new Exception(String.Format("Could not find property \"{0}\"", propertyName));
        }
```

```
        #endregion // Check property name
}
```

در این کلاس، ما پارامتر propertyName را از متد OnPropertyChanged، توسط صفت CallerMemberName حاشیه نویسی کرده‌ایم. این کار باعث می‌شود در Setterهای ویژگی‌ها، به راحتی بدون نوشتن نام ویژگی، عملیات اطلاع رسانی تغییرات را انجام دهیم. بدین صورت که کافیست متد OnPropertyChanged بدون هیچ آرگومانی در Setter فراخوانی شود و صفت CallerMemberName به صورت اتوماتیک نام ویژگی ای که فراخوانی از درون آن انجام شده است را درون پارامتر propertyName قرار می‌دهد.

پس کلاس PersonViewModel را به صورت زیر می‌توانیم اصلاح و تکمیل کنیم:

```
public class PersonViewModel : ViewModelBase
{
        private string firstName;
        public string FirstName
        {
            get { return firstName; }
            set
            {
                this.firstName = value;

                OnPropertyChanged();
                OnPropertyChanged(() => this.FullName);
            }
        }

        private string lastName;
        public string LastName
        {
            get { return lastName; }
            set
            {
                this.lastName = value;

                OnPropertyChanged();
                OnPropertyChanged(() => this.FullName);
            }
        }

        public string FullName
        {
            get { return string.Format("{0} {1}", FirstName, LastName); }
        }
}
```

همانطور که می‌بینید متد OnPropertyChanged بدون آرگومان فراخوانی میشود. اکنون اگر شما اقدام به Refactor کردن کد خود بکنید دیگر نگرانی از بابت تغییر نکردن رشته‌ها و کامنت‌ها نخواهید داشت و مطمئن هستید، نام ویژگی هر چیزی که باشد، به صورت خودکار به متد ارسال خواهد شد.

کلاس ViewModelBase یک پیاده سازی دیگر از OnPropetyChanged هم دارد که به شما اجازه می‌دهد با استفاده دستورات لامبدا، OnPropertyChanged را برای هر یک از اعضای دلخواه کلاس نیز فراخوانی کنید. همانطور که در مثال فوق می‌بینید، تغییرات نام خانوادگی در نام کامل شخص نیز اثرگذار است. در نتیجه به وسیله‌ی یک Func به راحتی بیان می‌کنیم که FullName هم تغییر کرده است و اطلاع رسانی برای آن نیز باید صورت پذیرد.

برای استفاده از صفت CallerMemberName باید دات نت هدف خود را 4.5 یا 4.6 قرار دهید.

ارجاع:

Raise INPC witout string name

نظرات خوانندگان

نویسنده: بهزاد
تاریخ: ۱۳۹۴/۰۴/۱۸ ۱۰:۲

با ایجاد Exception و خواندن StackTrace در نسخه‌های پایین‌تر هم میتوان به نام تابع فراخواننده پی برد