عنوان: آموزش WAF (مشاهده تغییرات خواص ViewModel در Controller)
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۱/۱۷ ۹:۳۵
آدرس: www.dotnettips.info
گروه‌ها: MVVM, WPF, WAF

قصد داریم در مثال پست قبلی برای Command مورد نظر، عملیات اعتبارسنجی را فعال کنیم. اگر با الگوی MVVM آشنایی داشته باشید می‌دانید که می‌توان برای Commandها اکشنی به عنوان CanExecute تعریف کرد و در آن عملیات اعتبارسنجی را انجام داد. اما از آن جا که پیاده سازی این روش زمانی مسیر است که تغییرات خواص ViewModel در دسترس باشد در نتیجه در WAF مکانیزمی جهت ردیابی تغییرات خواص ViewModel در کنترلر فراهم شده است. در نسخه‌های قبلی WAF (قبل از نسخه 3) هر کنترلر از کلاس پایه ای به نام Controller ارث می‌برد که متد هایی جهت ردیابی تغییرات در آن در نظر گرفته شده بود به صورت زیر:

```
public class MyController : Controller
    {
        [ImportingConstructor]
        public MyController(MyViewModel viewModel)
        {
            ViewModelCore = viewModel;
        }

        public MyViewModel ViewModelCore
        {
            get;
            private set;
        }

        public void Run()
        {
            AddWeakEventListener(ViewModelCore , ViewModelCoreChanged)
        }

        private void ViewModelCoreChanged(object sender , PropertyChangedEventArgs e)
        {
            if(e.PropertyName=="CurrentItem")
            {

            }
        }
    }
```

همان طور که مشاهده می‌کنید با استفاده از متد AddWeakEventListener توانستیم تمامی تغییرات خواص ViewModel مورد نظر را از طریق متد ViewModelCoreChanged ردیابی کنیم. این متد بر مبنای الگوی WeakEvent پیاده سازی شده است. البته این تغییرات فقط زمانی قابل ردیابی هستند که در ViewModel متد RaisePropertyChanged برای متد set خاصیت فراخوانی شده باشد.
از آنجا که در دات نت 4.5 یک پیاده سازی خاص از الگوی WeakEvent در کلاس PropertyChangedEventManager موجود در اسمبلی WindowsBase و فضای نام System.ComponentModel انجام شده است در نتیجه توسعه دهندگان این کتابخانه نیز تصمیم به استفاده از این روش گرفتند که نتیجه آن Obsolete شدن کلاس پایه کنترلر در نسخه‌های 3 به بعد آن است. در روش جدید کافیست به صورت زیر عمل نمایید:

```
[Export]
    public class BookController
    {
        [ImportingConstructor]
        public BookController(BookViewModel viewModel)
        {
            ViewModelCore = viewModel;
        }

        public BookViewModel ViewModelCore
        {
            get;
            private set;
        }

        public DelegateCommand RemoveItemCommand
```

```
        {
            get;
            private set;
        }

        private void ExecuteRemoveItemCommand()
        {
            ViewModelCore.Books.Remove(ViewModelCore.CurrentItem);
        }

        private bool CanExecuteRemoveItemCommand()
        {
            return ViewModelCore.CurrentItem != null;
        }
        private void Initialize()
        {
            RemoveItemCommand = new DelegateCommand(ExecuteRemoveItemCommand ,
CanExecuteRemoveItemCommand);
            ViewModelCore.RemoveItemCommand = RemoveItemCommand;
        }

        public void Run()
        {
            var result = new List<Book>();
            result.Add(new Book { Code = 1, Title = "Book1" });
            result.Add(new Book { Code = 2, Title = "Book2" });
            result.Add(new Book { Code = 3, Title = "Book3" });

            Initialize();
            ViewModelCore.Books = new ObservableCollection<Models.Book>(result);

            PropertyChangedEventManager.AddHandler(ViewModelCore, ViewModelChanged, "CurrentItem");

            (ViewModelCore.View as IBookView).Show();
        }

        private void ViewModelChanged(object sender,PropertyChangedEventArgs e)
        {
            if(e.PropertyName == "CurrentItem")
            {
                RemoveItemCommand.RaiseCanExecuteChanged();
            }
        }
    }
```

تغییرات:

»ابتدا متدی به نام CanExecuteRemoveItemCommand ایجاد کردیم و کدهای اعتبارسنجی را در آن قرار دادیم؛

»هنگام تعریف Command مربوطه متد بالا را به DelegateCommand رجیستر کردیم:

```
RemoveItemCommand = new DelegateCommand(ExecuteRemoveItemCommand , CanExecuteRemoveItemCommand);
```

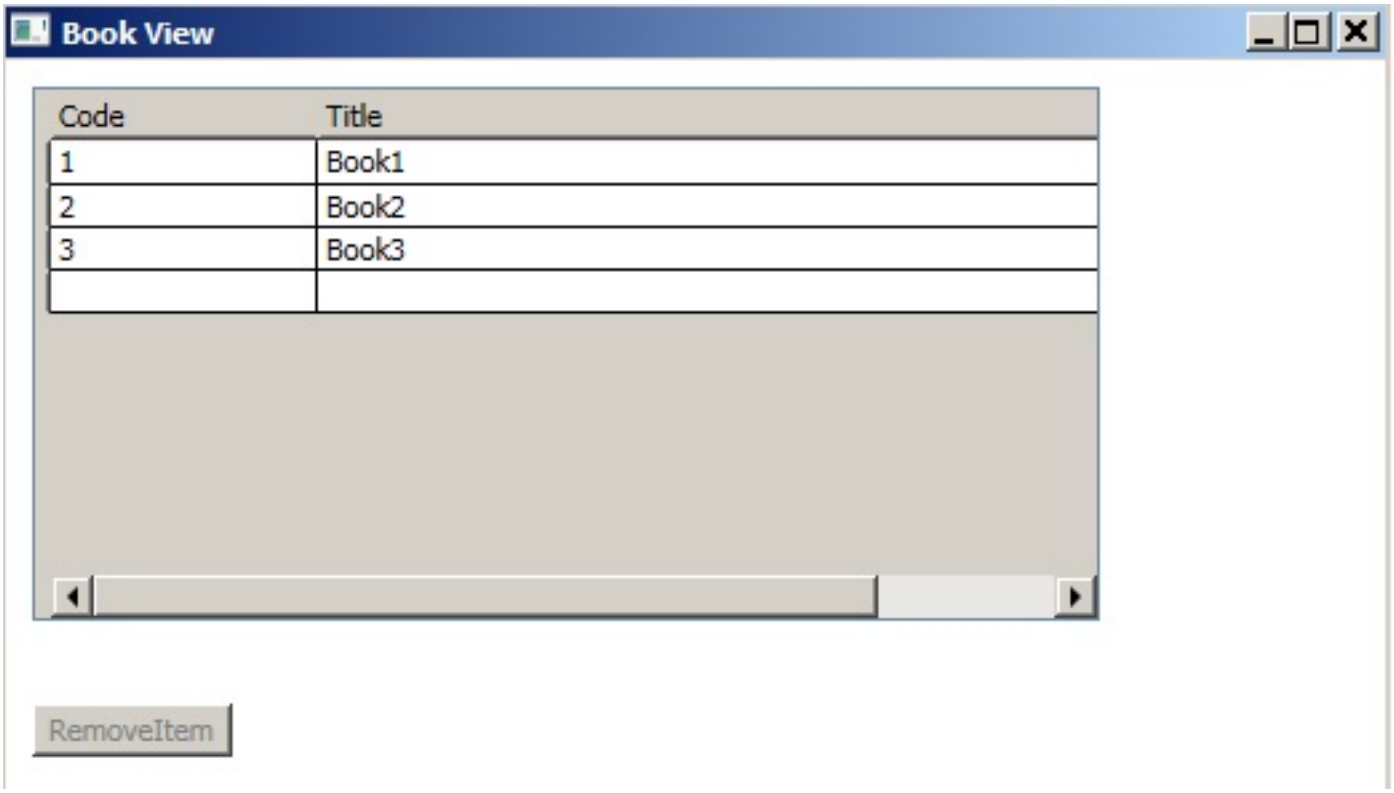
در این حالت بعد از اجرای برنامه همواره دکمه RemoveItem غیر فعال خواهد بود. دلیل آن این است که بعد از انتخاب آیتم مورد نظر از لیست باید کنترلر را متوجه تغییر در مقدار خاصیت CurrentItem نماییم. بدین منظور کد زیر را به متد Run اضافه کردم:

```
PropertyChangedEventManager.AddHandler(ViewModelCore, ViewModelChanged, "CurrentItem");
```

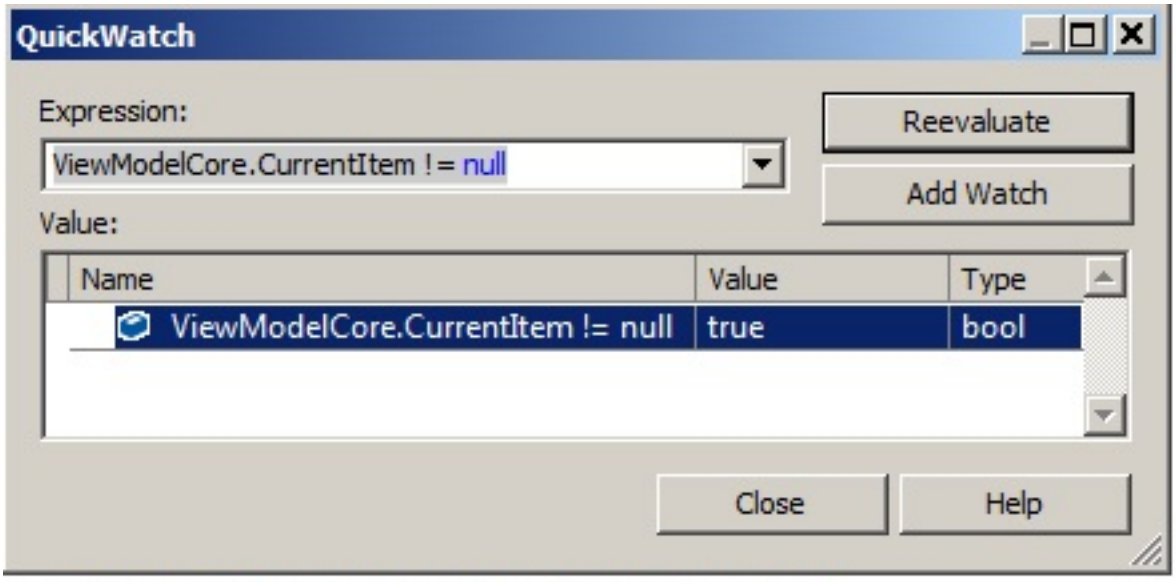
دستور بالا دقیقا معادل دستور AddWeakEventListener موجود در نسخه‌های قدیمی WAF است. سپس در صورتی که نام خاصیت مورد نظر CurrentItem بود با استفاده از دستور RaiseCanExecuteChanged در کلاس DelegateCommand کنترلر را ملزم به اجرای دوباره متد CanExecuteRemoveItemCommand می‌کنیم.

اجرای برنامه:

ابتدا دکمه RemoveItem غیر فعال است:

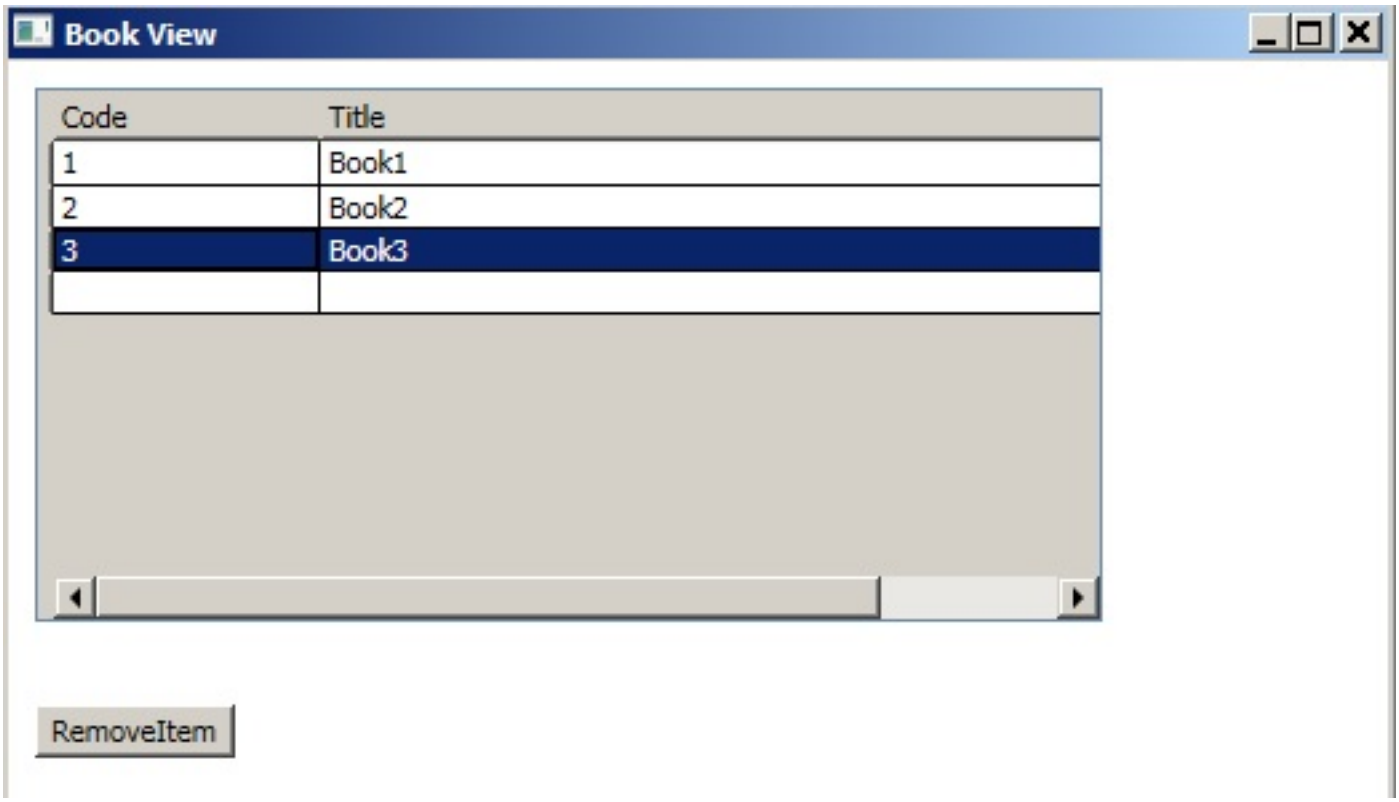


بعد از انتخاب یکی از گزینه و فراخوانی مجدد متد CanExecuteRemoveItemCommand دکمه مورد نظر فعال می‌شود:




```
private bool CanExecuteRemoveItemCommand()
{
    return ViewModelCore.CurrentItem != null;
}
```

و در نهایت دکمه RemoveItem فعال خواهد شد:



دانلود سورس پروژه