عنوان: **آشنایی با Refactoring - قسمت 6**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۱۷ ۰۰:۰۸:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Refactoring

در ادامه بحث «حذف کدهای تکراری»، روش Refactoring دیگری به نام " Extract Superclass " وجود دارد که البته در بین برنامه نویس‌های دات نت به نام Base class بیشتر مشهور است تا Superclass. هدف آن هم انتقال کدهای تکراری بین چند کلاس، به یک کلاس پایه و سپس ارث بری از آن می‌باشد.

یک مثال:
در WPF و Silverlight جهت مطلع سازی رابط کاربری از تغییرات حاصل شده در مقادیر داده‌ها، نیاز است کلاس مورد نظر، اینترفیس INotifyPropertyChanged را پیاده سازی کند:

```
using System.ComponentModel;

namespace Refactoring.Day6.ExtractSuperclass.Before
{
    public class User : 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));
        }
    }
}
```

و نکته‌ی مهم این است که اگر 100 کلاس هم داشته باشید، باید این کدهای تکراری اجباری مرتبط با raisePropertyChanged را در آن‌ها قرار دهید. به همین جهت مرسوم است برای کاهش حجم کدهای تکرای، قسمت‌های تکراری کد فوق را در یک کلاس پایه قرار می‌دهند:

```
using System.ComponentModel;

namespace Refactoring.Day6.ExtractSuperclass.After
{
    public class ViewModelBase : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        protected void RaisePropertyChanged(string propertyName)
        {
            var handler = PropertyChanged;
            if (handler == null) return;
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
```

```
    }
}
```

و سپس از آن ارث بری می‌کنند:

```
namespace Refactoring.Day6.ExtractSuperclass.After
{
    public class User : ViewModelBase
    {
        string _name;
        public string Name
        {
            get { return _name; }
            set
            {
                if (_name == value) return;
                _name = value;
                RaisePropertyChanged("Name");
            }
        }
    }
}
```

به این ترتیب این کلاس پایه در ده‌ها و صدها کلاس قابل استفاده خواهد بود، بدون اینکه مجبور شویم مرتبا یک سری کد تکراری «اجباری» را copy/paste کنیم.

مثالی دیگر:

اگر با ORM های Code first کار کنید، نیاز است تا ابتدا طراحی کار توسط کلاس‌های ساده دات نتی انجام شود؛ که اصطلاحا به آن‌ها POCO یا Plain old CLR objects یا Plain old .NET Classes هم گفته می‌شود. در بین این کلاس‌ها، متداول است که یک سری از خصوصیات، تکراری و مشترک باشد؛ مثلا تمام کلاس‌ها تاریخ ثبت رکورد را هم داشته باشند به همراه نام کاربر و مشخصاتی از این دست. اینجا هم برای حذف کدهای تکراری، یک Base class طراحی می‌شود: (+)

نظرات خوانندگان

نویسنده: A.Karimi
تاریخ: ۱۳۹۰/۰۷/۱۸ ۰۰:۱۶:۲۰

با عرض پوزش از اینکه مطلبی که می‌نویسم به پست بی ربط است. مایل بودم نظر شما را در مورد یک سوال، در صورتی که با RIA آشنایی دارید، بدانم که در stackoverflow مطرح کردم و پاسخی دریافت نکردم! سوال به طور خلاصه این است که «وقتی ما می‌خواهیم یک DTO پیچیده را به سمت سرور انتقال دهیم و در یک Round trip عملیات مورد نظرمان انجام شود (مثلاً یک Bulk insert یا چیزی شبیه آن) آیا در RIA راهی برای اینکار وجود دارد؟ یا اینکه باید از WCF Services سنتی در کنار RIA استفاده کنیم؟"

لینک StackOverflow:
http://stackoverflow.com/questions/7632337/send-custom-complex-objects-to-silverlight-ria-services

ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۱۸ ۰۹:۱۹:۰۵

علاوه بر مطالبی که اونطرف نوشتم، فورم اصلی RIA Services اینجا است: [^] . بگردید از این مورد زیاد دارد.

نویسنده: A.Karimi
تاریخ: ۱۳۹۰/۰۷/۱۸ ۱۹:۱۸:۰۹

از پیگیری شما متشکرم. نمی‌دانم شاید بی دقتی کردم اما گشت و گذار در این زمینه داشتم و متاسفانه چیزی پیدا نکردم. با استفاده از [Composition] مشکل متدهای اضافه Insert حل شد. اما هنوز برای Expose کردن شی اصی نیاز به یک متد Get یا Query دارم. آیا راحی وجود دارد که بدون نوشتن یک متد Get یا Query یک کلاس را با استفاده از RIA به سمت کلاینت Expose کرد؟ انتظار من یک Attribute برای DomainService بود ولی فعلاً چیزی پیدا نکرده‌ام.

ممنوم.

نویسنده: A.Karimi
تاریخ: ۱۳۹۰/۰۷/۱۸ ۱۹:۲۴:۱۶

البته منظور من از یک شی یا کلاس، یک کلاس است که به صورت دستی ساخته شده و نه کلاس‌های EF یا از این قبیل. امکان Expose کردن آنها به راحتی با استفاده از خصیصه‌ی LinqToEntitiesDomainServiceDescriptionProvider امکان پذیر است. اما در مورد یک Entity دست ساز چیزی نیافتم!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۱۸ ۲۲:۵۲:۲۷

این سایت در مورد RIA Services و DTO مطلب زیاد دارد. به مشکل مورد نظر شما هم اشاره کرده؛ در قسمت RIA and DTO - Part 2 : [^]

نویسنده: A.Karimi
تاریخ: ۱۳۹۰/۰۷/۲۲ ۱۹:۳۰:۰۸

ممنون. خیلی کمک کردید. البته این وبلاگهای مفید که اکثراً ف.ی.ل.ت.ر هستند و من به سختی توانستم مروری بکنم که متاسفانه چیزی در مورد Expose کردن Entityها با آن روشی که گفتم نیافتم البته باید با دقت بیشتری مرور کنم.

در هر صورت باز هم ممنون و عذر خواهی به علت بی ربط بودن کامنتها به پست.