عنوان: **آشنایی با Refactoring - قسمت 1**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۱۲ ۰۹:۴۶:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Refactoring

کارهای سورس باز قابل توجهی از برنامه نویس‌های ایرانی یافت نمی‌شوند؛ عموما کارهای ارائه شده در حد یک سری مثال یا کتابخانه‌های کوچک است و در همین حد. یا گاهی هم انگشت شمار پروژه‌هایی کامل. مثل یک وب سایت یا یک برنامه نصفه نیمه دبیرخانه و امثال آن. این‌ها هم خوب است از دیدگاه به اشتراک گذاری اطلاعات، ایده‌ها و هم ... یک مزیت دیگر را هم دارد و آن این است که بتوان کیفیت عمومی کد نویسی را حدس زد.
اگر کیفیت کدها رو بررسی کنید به یک نتیجه‌ی کلی خواهید رسید: "عموم برنامه نویس‌های ایرانی (حداقل این‌هایی که چند عدد کار سورس باز به اشتراک گذاشته‌اند) با مفهومی به نام Refactoring هیچگونه آشنایی ندارند". مثلا یک برنامه‌ی WinForm تهیه کرده‌اند و کل سورس برنامه همان چند عدد فرم برنامه است و هر فرم بالای 3000 سطر کد دارد. دوستان عزیز! به این می‌گویند «فاجعه‌ای به نام کدنویسی!» صاحب اول و آخر این نوع کدها خودتان هستید! شاید به همین جهت باشد که عمده‌ی پروژه‌های سورس باز پس از اینکه برنامه نویس اصلی از توسعه‌ی آن دست می‌کشد، «می‌میرند». چون کسی جرات نمی‌کند به این کدها دست بزند. مشخص نیست الان این قسمت را که تغییر دادم، کجای برنامه به هم ریخت. تستی ندارند. ساختاری را نمی‌توان از آن‌ها دریافت. منطق قسمت‌های مختلف برنامه از هم جدا نشده است. برنامه یک فرم است با چند هزار سطر کد در یک فایل! کار شما شبیه به کد اسمبلی چند هزار سطری حاصل از decompile یک برنامه که نباید باشد!
به همین جهت قصد دارم یک سری «ساده» Refactoring را در این سایت ارائه دهم. روی سادگی هم تاکید کردم، چون اگر عموم برنامه نویس‌ها با همین موارد به ظاهر ساده آشنایی داشتند، کیفیت کد نویسی بهتری را می‌شد در نتایج عمومی شده، شاهد بود.
این مورد در راستای نظر سنجی انجام شده هم هست؛ درخواست مقالات خالص سی شارپ در صدر آمار فعلی قرار دارد.

Refactoring چیست؟

Refactoring به معنای بهبود پیوسته کیفیت کدهای نوشته شده در طی زمان است؛ بدون ایجاد تغییری در عملکرد اصلی برنامه. به این ترتیب به کدهایی دست خواهیم یافت که قابلیت آزمون پذیری بهتری داشته، در مقابل تغییرات مقاوم و شکننده نیستند و همچنین امکان به اشتراک گذاری قسمت‌هایی از آن‌ها در پروژه‌های دیگر نیز میسر می‌شود.

قسمت اول - مجموعه‌ها را کپسوله کنید

برای مثال کلاس‌های ساده زیر را در نظر بگیرید:

```
namespace Refactoring.Day1.EncapsulateCollection
{
    public class OrderItem
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public int Amount { set; get; }
    }
}
```

```
using System.Collections.Generic;

namespace Refactoring.Day1.EncapsulateCollection
{
    public class Orders
    {
        public List<OrderItem> OrderItems { set; get; }
    }
```

```
}
```

نکته اول: هر کلاس باید در داخل یک فایل جدا قرار گیرد. «لطفا» یک فایل درست نکنید با 50 کلاس داخل آن. البته اگر باز هم یک فایل باشد که بتوان 50 کلاس را داخل آن مشاهده کرد که چقدر هم عالی! نه اینکه یک فایل باشد تا بعدا 50 کلاس را با Refactoring از داخل آن بیرون کشید!

قطعه کد فوق، یکی از روش‌های مرسوم کد نویسی است. مجموعه‌ای به صورت یک List عمومی در اختیار مصرف کننده قرار گرفته است. حال اجازه دهید تا با استفاده از برنامه FxCop برنامه فوق را آنالیز کنیم. یکی از خطاهایی را که نمایش خواهد داد عبارت زیر است:

```
Error, Certainty 95, for Do Not Expose Generic Lists
```

بله. لیست‌های جنریک را نباید به همین شکل در اختیار مصرف کننده قرار داد؛ چون به این صورت هر کاری را می‌توانند با آن انجام دهند، مثلا کل آن را تعویض کنند، بدون اینکه کلاس تعریف کننده آن از این تغییرات مطلع شود.
پیشنهاد FxCop این است که بجای List از Collection یا IList و موارد مشابه استفاده شود. اگر اینکار را انجام دهیم اینبار به خطای زیر خواهیم رسید:

```
Warning, Certainty 75, for Collection Properties Should Be ReadOnly
```

FxCop پیشنهاد می‌دهد که مجموعه تعریف شده باید فقط خواندنی باشد.

چکار باید کرد؟
بجای استفاده از List جهت ارائه مجموعه‌ها، از IEnumerable استفاده کنید و اینبار متدهای Add و Remove اشیاء به آن‌را به صورت دستی تعریف نمائید تا بتوان از تغییرات انجام شده بر روی مجموعه ارائه شده، در کلاس اصلی آن مطلع شد و امکان تعویض کلی آن‌را از مصرف کننده گرفت. برای مثال:

```
using System.Linq;
using System.Collections.Generic;

namespace Refactoring.Day1.EncapsulateCollection
{
    public class Orders
    {
        private int _orderTotal;
        private List<OrderItem> _orderItems;

        public IEnumerable<OrderItem> OrderItems
        {
            get { return _orderItems; }
        }

        public void AddOrderItem(OrderItem orderItem)
        {
            _orderTotal += orderItem.Amount;
            _orderItems.Add(orderItem);
        }

        public void RemoveOrderItem(OrderItem orderItem)
        {
            var order = _orderItems.Find(o => o == orderItem);
            if (order == null) return;
```

```
            _orderTotal -= orderItem.Amount;
            _orderItems.Remove(orderItem);
        }
    }
}
```

اکنون اگر برنامه را مجددا با fxCop آنالیز کنیم، دو خطای ذکر شده دیگر وجود نخواهند داشت. اگر این تغییرات صورت نمی‌گرفت، امکان داشتن فیلد _orderTotal غیر معتبری در کلاس Orders به شدت بالا می‌رفت. زیرا مصرف کننده مجموعه OrderItems می‌توانست به سادگی آیتمی را به آن اضافه یا از آن حذف کند، بدون اینکه کلاس Orders از آن مطلع شود یا اینکه بتواند عکس العمل خاصی را بروز دهد.

نظرات خوانندگان

نویسنده: mrdotnet
تاریخ: ۱۳۹۰/۰۷/۱۲ ۱۳:۲۶:۰۸

سلام
با توجه به اینکه نسخه جدید fxCop با Windows SDK ارائه شده که حجم SDK حدود 600 مگ هست ، دوستانی که به هر دلیلی مایل به دانلود کل SDK نیستند میتونن از فایل زیر به در یافت تنها fxCop با حجم 10 مگ اقدام کنند.
http://www.mediafire.com/?hq3k13d7cuoxe7rدر ضمن یک آموزش نحوه استفاده مختصر و مفید از این ابزار رو میتونید در این لینک ببینید http://www.codeproject.com/KB/dotnet/FxCop.aspx

حالا شما آماده هستید تا سری آموزش های آشنایی با Refactoring رو دنبال کنید.

نویسنده: Tohid Azizi
تاریخ: ۱۳۹۰/۰۹/۰۳ ۲۰:۱۱:۴۷

با سلام و تشکر از سری مقالات بسیار مفید ریفکتورینگ.
در مورد خطای «Do Not Expose Generic Lists» و کد ریفکتور شده ی آن، آیا راهی وجود دارد که بتوان از قابلیت های اندکس ICollection برای پروپرتی استفاده کرد اما نتوان با استفاده از Add یا Insert عضوی به آن اضافه کرد؟مثلا ـ طبق مثال شما ـ داشته باشیم:(for (int i=0; i<Orders.OrderItems.Count; i++) Console.WriteLine(Orders.OrderItems[i].Price;یا حتا:for (int i=0; i<Orders.OrderItems.Count; i++) Orders.OrderItems[i].Tax = Orders.OrderItems[i].Price * .05;اما نتوان نمونه ی جدیدی به لیست OrderItems اضافه کرد؟Orders.OrderItems.Add(newOrderItem); //raise errorبا تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۰۳ ۲۱:۱۳:۵۲

می‌تونید چیزی شبیه به ReadOnlyCollection درست کنید ((^)). ReadOnlyCollection در حقیقت ICollection را پیاده سازی کرده با این تفاوت که پیاده سازی متد Add آن را معادل throw NotSupportedException قرار داده ((^)).

نویسنده: سید ایوب کوکبی
تاریخ: ۱۳۹۲/۰۹/۱۳ ۱۴:۱۸

مبحث مهمی رو دارید پیش میبرید، امیدوارم به قسمت‌های پیشرفته و حساس‌تر هم برسیم، همچنین تجربیات احتمالی خودتون رو هم دخیل در توضیحات کنید تا اهمیت مبحث مطروحه دو چندان بشه!
ممنونم.

نویسنده: سید ایوب کوکبی
تاریخ: ۱۳۹۲/۰۹/۱۳ ۱۴:۳۷

سلام، اگه میشه جای دیگه ای آپلود کنید، لینک خرابه! ممنونم/.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۳ ۱۴:۳۹

تکمیل شده این مبحث را در برچسب refactoring در طی 14 قسمت می‌توانید پیگیری کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۳ ۱۴:۵۶

- 2 سال قبل آپلود شده بوده.
- از اینجا دریافت کنید:

FxCop10.7z