عنوان: **آشنایی با Refactoring - قسمت 10**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۸ ۰۰:۲۰:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Refactoring

یکی دیگر از روش‌هایی که جهت بهبود کیفیت کدها مورد استفاده قرار می‌گیرد، « طراحی با قراردادها » است؛ به این معنا که «بهتر است» متدهای تعریف شده پیش از استفاده از آرگومان‌های خود، آن‌ها را دقیقا بررسی کنند و به این نوع پیش شرط‌ها، قرارداد هم گفته می‌شود.
نمونه‌ای از آن‌را در قسمت 9 مشاهده کردید که در آن اگر آرگومان‌های متد AddRole، خالی یا نال باشند، یک استثناء صادر می‌شود. این نوع پیغام‌های واضح و دقیق در مورد عدم اعتبار ورودی‌های دریافتی، بهتر است از پیغام‌های کلی و نامفهوم null reference exception که بدون بررسی stack trace و سایر ملاحظات، علت بروز آن‌ها مشخص نمی‌شوند.
در دات نت 4، جهت سهولت این نوع بررسی‌ها، مفهوم Code Contracts ارائه شده است. (این نام هم از این جهت بکارگرفته شده که Design by Contract نام تجاری شرکت ثبت شده‌ای در آمریکا است!)

یک مثال:
متد زیر را در نظر بگیرید. اگر divisor مساوی صفر باشد، استثنای کلی DivideByZeroException صادر می‌شود:

```
namespace Refactoring.Day10.DesignByContract.Before
{
    public class MathMehods
    {
        public double Divide(int dividend, int divisor)
        {
            return dividend / divisor;
        }
    }
}
```

روش متداول «طراحی با قراردادها» جهت بهبود کیفیت کد فوق پیش از دات نت 4 به صورت زیر است:

```
using System;

namespace Refactoring.Day10.DesignByContract.After
{
    public class MathMehods
    {
        public double Divide(int dividend, int divisor)
        {
            if (divisor == 0)
                throw new ArgumentException("divisor cannot be zero", "divisor");

            return dividend / divisor;
        }
    }
}
```

در اینجا پس از بررسی آرگومان divisor، قرارداد خود را به آن اعمال خواهیم کرد. همچنین در استثنای تعریف شده، پیغام واضح‌تری به همراه نام آرگومان مورد نظر، ذکر شده است که از هر لحاظ نسبت به استثنای استاندارد و کلی DivideByZeroException مفهوم‌تر است.

در دات نت 4 ، به کمک امکانات مهیای در فضای نام System.Diagnostics.Contracts، این نوع بررسی‌ها نام و امکانات درخور

خود را یافته‌اند:

```
using System.Diagnostics.Contracts;

namespace Refactoring.Day10.DesignByContract.After
{
    public class MathMehods
    {
        public double Divide(int dividend, int divisor)
        {
            Contract.Requires(divisor != 0, "divisor cannot be zero");

            return dividend / divisor;
        }
    }
}
```

البته اگر قطعه کد فوق را به همراه divisor=0 اجرا کنید، هیچ پیغام خاصی را مشاهده نخواهید کرد؛ از این لحاظ که نیاز است تا فایل‌های مرتبط با آن‌را از این آدرس دریافت و نصب کنید. این کتابخانه با VS2008 و VS2010 سازگار است. پس از آن، برگه‌ی Code contracts به عنوان یکی از برگه‌های خواص پروژه در دسترس خواهد بود و به کمک آن می‌توان مشخص کرد که برنامه حین رسیدن به این نوع بررسی‌ها چه عکس العملی را باید بروز دهد.

برای مطالعه بیشتر:

Code Contracts in C#

Code Contracts in .NET 4

Introduction to Code Contracts

نظرات خوانندگان

نویسنده: Nima
تاریخ: ۱۳۹۰/۰۷/۳۰ ۱۰:۰۹:۴۹

سلام آقای نصیری
با تشکر از مطالب بسیار ارزنده شما. من در مورد Code Contract تحقیق کردم و سعی کردم یکم باهاش کار کنم. اول اینکه واقعا دلیل این رو نمیدونم که چرا باید ما یک برنامه خارجی را نصب کنیم تا این کدها واکنش نشان بدن.دوم اینکه نمیشه از این کدها داخل بلاک Try-Catch استفاده کرد. و میخواستم نظر شما را راجع به http://fluentvalidation.codeplex.com/ بدونم. این فریم ورک تقریبا همون کار رو انجام میده ولی استفاده ازش راحتتره. در کل به نظرم در بحث Refactoring به یک نقطه حساس رسیدیم و اون Validation هست. به نظر حقیر مسئله Validation و حلش میتونه سهم به سزایی در خوانایی کد داشته باشه.
ممنون میشم اگر مثل همیشه ما رو از نظرات ارزشمند خودتون مستفیض کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۳۰ ۱۱:۵۵:۲۶

- در مورد طراحی آن اگر نظری دارید لطفا به تیم BCL اطلاع دهید: http://blogs.msdn.com/b/bclteam/
- بحث code contacts در اینجا فراتر است از validation متداول. این نوع اعتبارسنجی‌های متداول عموما و در اکثر موارد جهت بررسی preconditions هستند؛ در حالیکه اینجا post-conditions را هم شامل می‌شوند.
- در مورد کتابخانه‌های Validation هر کسی راه و روش خاص خودش را دارد. یکی ممکن است از DataAnnotations خود دات نت استفاده کند (و http://xval.codeplex.com/)، یکی از http://validationframework.codeplex.com/ یا از http://code.google.com/p/kosher/ و یا http://tnvalidate.codeplex.com/ و یا حتی NHibernate هم کتابخانه اعتبارسنجی خاص خودش را دارد.
در کل هدف این است که این کار بهتر است انجام شود. حالا با هر کدام که راحت هستید. مانند وجود انواع فریم‌ورک‌های Unit test یا انواع مختلف سورس کنترل‌ها. مهم این است که از یکی استفاده کنید.