عنوان: **refactoring کدهای تکراری به کمک Actionها**
نویسنده: محمد رضا منشادی
تاریخ: ۱۳۹۲/۰۸/۱۳ ۱:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: Refactoring

در برخی از مواقع بر روی اشیاء یک لیست، در یک کلاس، با استفاده از حلقه‌های foreach یا for کارهای متفاوتی انجام می‌شود. به عنوان مثال در یک لیست که از سطرهای فاکتور تشکیل شده است، می‌خواهیم جمع مقادیر کلیه سطرهای فاکتور یا جمع مبلغ یا مالیات یا تخفیف آنها را بدست آوریم . با وجود سادگی حلقه‌های foreach و for، ممکن است که در برخی از مواقع از راه متفاوتی استفاده شود. برای مثال اجازه بدهید مثال ذیل را با هم بررسی کنیم:

در کلاس Invoice دو متد وجود دارد با نام های CalculateTotalTax و CalculateTotal

متد CalculateTotalTax مجموع مالیات و متد CalculateTotal مجموع مقدار این فاکتور را بدست می‌آورد .

```
public float CalculateTotalTax1()
        {
            IList<InvoiceLineItem> invoiceLineItem = new List<InvoiceLineItem>();
            Decimal result = 0M;
            foreach (InvoiceLineItem index in invoiceLineItem)
            {
                result += (Decimal)index.CalculateTax();
            }
            return (float)result;
        }


        public float CalculateTotal()
        {
            IList<InvoiceLineItem> invoiceLineItem = new List<InvoiceLineItem>();
            Decimal result = 0M;
            foreach (InvoiceLineItem index in invoiceLineItem)
            {
                result += (Decimal)index.CalculateSubTotal();
            }
            return (float)result;
        }
```

این دو متد به طور عمومی یک چیز را دارند: هر دو کارهای متفاوتی را بر روی لیست سطرهای فاکتور انجام می‌دهند.

ما می‌توانیم مسئولیت چرخش در لیست سطرهای فاکتور را از این متدها برداریم و آن را از IEnumerable جدا کنیم؛ به وسیله ایجاد یک متد که پارامتر ورودی Action<T> delegate را دریافت می‌کند و این delegate را برای هر سطر در هر چرخش اجرا می‌کند.

```
public void PerformActionOnAllLineItems(Action<InvoiceLineItem> action)
        {
            IList<InvoiceLineItem> invoiceLineItem = new List<InvoiceLineItem>();

            invoiceLineItem.Add(new InvoiceLineItem { Id = 1, amount = 10, Price = 10000 });
            invoiceLineItem.Add(new InvoiceLineItem { Id = 2, amount = 10, Price = 10000 });
            invoiceLineItem.Add(new InvoiceLineItem { Id = 3, amount = 10, Price = 10000 });
            invoiceLineItem.Add(new InvoiceLineItem { Id = 4, amount = 10, Price = 10000 });

            foreach (InvoiceLineItem index in invoiceLineItem)
            {
                action(index);
            }
        }
```

و همچنین می‌توانیم دو متد خود را به شکل ذیل تغییر دهیم

```
float CalculateTotal()
        {
            Decimal result = 0M;
            PerformActionOnAllLineItems(delegate(InvoiceLineItem ili)
            {
```

```
                result += (Decimal)ili.CalculateSubTotal();
            });
            return (float)result;
        }
        float CalculateTotalTax()
        {
            Decimal result = 0M;
            PerformActionOnAllLineItems(delegate(InvoiceLineItem ili)
            {
                result += (Decimal)ili.CalculateTax();
            });
            return (float)result;
        }
```

منبع : کتاب Refactoring with Microsoft Visual Studio 2010

سورس نمونه : Advancedduplicatecoderefactoring.rar

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۰۸/۱۳ ۹:۳۵

ممنون از مطلبتون. از کجا می‌تونم کتابی رو که به عنوان منبع معرفی کردید، پیدا کنم؟
اصلا نسخه PDF اش هست؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۱۳ ۱۵:۱۰

بله. میشه از گوگل هم کمک گرفت. همون اولین یا حداکثر دومین لینک حاصل کافی است.