عنوان: **آشنایی با Refactoring - قسمت 12**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۰۵ ۲۳:۵۶:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Refactoring

قبلا در مورد تبدیل switch statement به الگوی استراتژی مطلبی را در این سایت مطالعه کرده‌اید (^) و بیشتر مربوط است به حالتی که داخل هر یک از case های یک switch statement چندین و چند سطر کد و یا فراخوانی یک تابع وجود دارد. حالت ساده‌تری هم برای refactoring یک عبارت switch وجود دارد و آن هم زمانی است که هر case، تنها از یک سطر تشکیل می‌شود؛ مانند:

```
namespace Refactoring.Day12.RefactoringSwitchStatement.Before
{
    public class Translator
    {
        public string ToPersian(string englishWord)
        {
            switch (englishWord)
            {
                case "zero":
                    return "صفر";
                case "one":
                    return "یک";
                default:
                    return string.Empty;
            }
        }
    }
}
```

در اینجا می‌توان از امکانات ساختار داده‌های توکار دات نت استفاده کرد و این switch statement را به یک dictionary تبدیل نمود:

```
using System.Collections.Generic;

namespace Refactoring.Day12.RefactoringSwitchStatement.After
{
    public class Translator
    {
        IDictionary<string, string> Words = new Dictionary<string, string>
        {
            { "zero", "صفر" },
            { "one", "یک" }
        };

        public string ToPersian(string englishWord)
        {
            string persianWord;
            if (Words.TryGetValue(englishWord, out persianWord))
            {
                return persianWord;
            }

            return string.Empty;
        }
    }
}
```

همانطور که ملاحظه می‌کنید هر case به یک key و هر return به یک value در Dictionary تعریف شده، تبدیل گشته‌اند. در اینجا هم بهتر است از متد TryGetValue جهت دریافت مقدار کلیدها استفاده شود؛ زیرا در صورت فراخوانی یک Dictionary با کلیدی که در آن موجود نباشد یک استثناء بروز خواهد کرد.
برای حذف این متد TryGetValue، می‌توان یک enum را بجای کلیدهای تعریف شده، معرفی کرد. به صورت زیر:

```
using System.Collections.Generic;

namespace Refactoring.Day12.RefactoringSwitchStatement.After
{
    public enum EnglishWord
    {
        Zero,
        One
    }

    public class Translator2
    {
        IDictionary<EnglishWord, string> Words = new Dictionary<EnglishWord, string>
        {
            { EnglishWord.Zero, "صفر" },
            { EnglishWord.One, "یک" }
        };

        public string ToPersian(EnglishWord englishWord)
        {
            return Words[englishWord];
        }
    }
}
```

به این ترتیب از یک خروجی پر از if و else و switch به یک خروجی ساده و بدون وجود هیچ شرطی رسیده‌ایم.

نظرات خوانندگان

نویسنده: afsharm
تاریخ: ۱۳۹۰/۰۸/۰۶ ۱۰:۴۸:۲۳

این یکی خیلی جالب بود

نویسنده: Farhad Yazdan-Panah
تاریخ: ۱۳۹۰/۰۸/۰۶ ۱۳:۱۰:۱۷

بسیار عالی. در معماری پروسسورهای اینتل دستوری برای کاری مشابه وجود دارد (فکر کنم یه چیزی به نام XLAT یا شبیهش). به این صورت که شما یک Look-up Table می سازید و همانند Dictionary در اینجا عمل می کنه.
تجربه شخصیم (در حد اسمبلی ماشین های X86) این روش سرعت بسیار بالاتری از حالت شرطی (مبتنی بر if) داره. در مورد Switch مطمئن نیستم.
در کل ممنون دارید کلی کیفیت کد نویسی ملتو بالا می برید.
اگه 10 تا وبلاگه دیگه مثل http://www.dotnettips.info بود الان ما وضع خیلی بهتری داشتیم.