عنوان: **آشنایی با Refactoring - قسمت 7**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۱۸ ۰۰:۰۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Refactoring

یکی دیگر از روش‌های Refactoring ، معرفی کردن یک کلاس بجای پارامترها است. عموما تعریف متدهایی با بیش از 5 پارامتر مزموم است:

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

namespace Refactoring.Day7.IntroduceParameterObject.Before
{
    public class Registration
    {
        public void Create(string name, DateTime date, DateTime validUntil,
                           IEnumerable<string> courses, decimal credits)
        {
            // do work
        }
    }
}
```

در این حالت بجای تعریف این تعداد بالای پارامترهای مورد نیاز، تمام آن‌ها را تبدیل به یک کلاس کرده و استفاده می‌کنند:

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

namespace Refactoring.Day7.IntroduceParameterObject.After
{
    public class RegistrationContext
    {
        public string Name {set;get;}
        public DateTime Date {set;get;}
        public DateTime ValidUntil {set;get;}
        public IEnumerable<string> Courses {set;get;}
        public decimal Credits { set; get; }
    }
}
```

```
namespace Refactoring.Day7.IntroduceParameterObject.After
{
    public class Registration
    {
        public void Create(RegistrationContext registrationContext)
        {
            // do work
        }
    }
}
```

یکی از مزایای این روش، منعطف شدن معرفی متدها است؛ به این صورت که اگر نیاز به افزودن پارامتر دیگری باشد، تنها کافی است یک خاصیت جدید به کلاس RegistrationContext اضافه شود و امضای متد Create، ثابت باقی خواهد ماند.

روش دیگر تشخیص نیاز به این نوع Refactoring ، یافتن پارامترهایی هستند که در یک گروه قرار می‌گیرند. برای مثال:

```
public int GetIndex(int pageSize, int pageNumber, ...) { ...
```

همانطور که ملاحظه می‌کنید تعدادی از پارامترها در اینجا با کلمه page شروع شده‌اند. بهتر است این پارامترهای مرتبط را به یک کلاس مجزا به نام Page انتقال داد.

نظرات خوانندگان

نویسنده: Farhad Yazdan-Panah
تاریخ: ۱۳۹۰/۰۷/۱۹ ۱۶:۵۳:۴۳

البته به نظر من در زمانیکه تابع مورد نظر یک گزارش (یا بخش از آن) باشد بهتره که استثنا قائل شد.فقط یک سوال: در حالاتیکه از کنترل هایی مثل ObjectDataSource استفاده بشه و بخواهیم یکی از این توابع (با ورودی جدید) را فراخوانی کنیم باید پیچیدگی زیادی به برنامه اضافه بشه (Serialize , ...). آیا چاره ای وجود دارد؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۱۹ ۱۷:۱۴:۰۳

بحث Refactoring در مورد طراحی کارهای شما معنا پیدا می‌کند؛ وگرنه اگر کتابخانه‌ی بسته دیگری، نیازهای خاص خودش را دیکته می‌کند، بدیهی است دست شما آنچنان باز نخواهد بود.
در مورد مطلبی که گفتید، بله می‌شود. در این حالت باید DataObject TypeName مربوط به ObjectDataSource را مشخص کنید: [^]
اگر می‌خواهید واقعا این اصول شیءگرایی را رعایت کنید، بهتر است به ASP.NET MVC کوچ کنید. Model binder آن، خودش به صورت خودکار این موارد را پوشش می‌دهد. نگارش بعدی ASP.NET Webforms هم کمی تا قسمتی از این Model binder رو به ارث برده ولی نه آنچنان که یک strongly typed view رو بتونید باهاش 100 درصد مثل MVC تعریف کنید.
در کل معماری ASP.NET Webforms مربوط به روزهای اول دات نت است و به نظر هم قرار نیست آنچنان تغییری بکند. به همین جهت MVC رو این وسط معرفی کرده‌اند.

نویسنده: Farhad Yazdan-Panah
تاریخ: ۱۳۹۰/۰۷/۱۹ ۲۲:۲۷:۰۸

ممنون از توضیحات.
در مورد جمله "البته به نظر من در زمانیکه تابع مورد نظر یک گزارش (یا بخش از آن) باشد بهتره که استثنا قائل شد." منظور من بخش ها و توابعی هستند که ما برای گزارشات استفاده می کنیم. (استخراج تعداد دانشجویان بر اساس بازه تولد، جنسیت، کلمه کلیدی از نام و .. و ...).
در این گونه موارد چون این تابع فقط یک بار و یک جا استفاده می شود آیا استفاده از این رویه کمی دست و پاگیر نیست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۱۹ ۲۲:۵۴:۲۸

خیر. اگر کمی با الگوهای MVC ، MVVM و امثال آن کار کنید، تهیه مدل جهت این موارد برای شما عادی خواهد شد. چون مجبورید که این‌ها را با حداقل یک کلاس مدل کنید.