عنوان: **نگاهی به انواع Aspects موجود در کتابخانه PostSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۱ ۲۰:۱۴
آدرس: www.dotnettips.info
گروه‌ها: Design patterns, AOP, C#, Architecture

تعدادی Aspect توکار در کتابخانه PostSharp قرار دارند که نقطه آغازین کار با آن‌را تشکیل می‌دهند. نمونه‌ای از آن‌را در قسمت قبل به نام OnMethodBoundaryAspect بررسی کردیم. اغلب این‌ها کلاس‌هایی هستند Abstract که با تهیه‌ی کلاس‌هایی مشتق شده از آن‌ها و override نمودن متدهای کلاس پایه، می‌توان Aspect جدیدی را ایجاد نمود. تمام این نوع Aspects در حقیقت نوعی مزین کننده به شمار می‌روند. در ادامه قصد داریم نگاهی داشته باشیم به سایر Aspects مهیای در کتابخانه PostSharp.

### 1) OnExceptionAspect

از OnExceptionAspect برای مدیریت استثناءهای متدها استفاده می‌شود. کار این Aspect، اضافه کردن try/catch به کدهای یک متد است و سپس فراخوانی متد OnException در صورت بروز خطایی در این بین.

```
using System;
using System.Reflection;
using PostSharp.Aspects;

namespace AOP03
{
    public class ApplicationExceptionHandlerAspect : OnExceptionAspect
    {
        public override void OnException(MethodExecutionArgs args)
        {
            Console.WriteLine("Exception Type: {0}, StackTrace: {1}",
                              args.Exception.GetType().Name,
                              args.Exception.StackTrace);
        }

        public override Type GetExceptionType(MethodBase targetMethod)
        {
            return typeof(ApplicationException);
        }
    }
}
```

مثالی را در این زمینه در کدهای فوق ملاحظه می‌کنید. اگر تنها متد OnException تحریف شود، try/catch خودکار اضافه شده به کدها، هر نوع استثنایی را مدیریت خواهد کرد. اما اگر متد GetExceptionType نیز در این بین مقدار دهی گردد، بر اساس نوع استثنای تعریف شده، کار فیلتر استثناها انجام می‌پذیرد و از مابقی صرفنظر خواهد شد.
نحوه استفاده از این Aspect نیز همانند مثال قسمت قبل است و جزئیات آن تفاوتی نمی‌کند.

### 2) LocationInterceptionAspect

این Aspect برخلاف سایر Aspectهایی که تاکنون بررسی کردیم، تنها در سطح خواص و فیلدهای یک کلاس عمل می‌کند. کار Interception در اینجا به معنای تحت کنترل قرار دادن اعمال set (پیش از فراخوانی set) و get (پیش از بازگشت مقدار) این خواص عمومی و حتی خصوصی تعریف شده است. کلمه Location در این Aspect به معنای متادیتای زمینه کاری است؛ مانند Name و FullName خواصی که مشغول به کار با آن‌ها هستیم.

```
using System;
using PostSharp.Aspects;

namespace AOP03
{
    public class ObjectInitializationAspect : LocationInterceptionAspect
    {
        public override void OnGetValue(LocationInterceptionArgs args)
        {
            if (args.GetCurrentValue() == null)
            {
                Console.WriteLine("Property {0} is null.", args.LocationFullName);
```

```
            }
        }
    }
}
```

یک نمونه از کاربرد آن‌را در مثال فوق مشاهده می‌کنید. در اینجا با تحریف متد OnGetValue، پیش از بازگشت مقداری از یک خاصیت، بررسی می‌شود که آیا مقدار آن null است یا خیر.
برای استفاده از آن نیز کافی است تا ویژگی ObjectInitializationAspect به خاصیتی دلخواه اضافه شود.
در اینجا 4 متد args.GetCurrentValue برای دریافت مقدار جاری خاصیت، args.SetNewValue جهت تنظیم مقداری جدید، args.ProceedGetValue و args.ProceedSetValue سبب اجرای حالت‌های get و set می‌شوند (چیزی شبیه به عملکرد اینترفیس IInterceptor که در قسمت‌های قبلی بررسی کردیم).

**3) EventInterceptionAspect**

EventInterceptionAspect همانطور که از نام آن نیز پیدا است، در سطح رخدادهای یک کلاس عمل می‌کند. سه متدی که این کلاس پایه برای تحت نظر قرار دادن اعمال رویدادگردان‌های یک کلاس در اختیار ما قرار می‌دهند شامل OnAddHandler، OnRemoveHandler و OnInvokeHandler هستند.

```
using PostSharp.Aspects;
using System;

namespace AOP03
{
    public class LogEventAspect : EventInterceptionAspect
    {
        public override void OnAddHandler(EventInterceptionArgs args)
        {
            Console.WriteLine("Event {0} added", args.Event.Name);
            args.ProceedAddHandler();
        }

        public override void OnRemoveHandler(EventInterceptionArgs args)
        {
            Console.WriteLine("Event {0} removed", args.Event.Name);
            args.ProceedRemoveHandler();
        }

        public override void OnInvokeHandler(EventInterceptionArgs args)
        {
            Console.WriteLine("Event {0} invoked", args.Event.Name);
            args.ProceedInvokeHandler();
        }
    }
}
```

مثالی را از نحوه تعریف یک EventInterceptionAspect مشاهده می‌کنید. در تمام حالاتی که متدهای کلاس پایه تحریف شده‌اند نیاز است از متدهای Proceed متناظر نیز استفاده شود تا برای مثال اضافه شدن، حذف و یا اجرای یک رویداد رخ دهند.

**مدیریت اعمال Aspects در زمان کامپایل**

یکی از متدهایی که در کلیه Aspects توکار فوق قابل تحریف است، CompileTimeValidate نام دارد.

```
public class LoggingAspect : OnMethodBoundaryAspect
    {
        public override bool CompileTimeValidate(System.Reflection.MethodBase method)
        {
            return !method.IsStatic;
        }
```

برای نمونه اگر آن‌را به OnMethodBoundaryAspect پیاده سازی شده در قسمت قبل، با تعاریف فوق اعمال کنیم، این Aspect

سفارشی دیگر به متدهای استاتیک، اعمال نخواهد شد. به این ترتیب می‌توان بر روی نحوه کامپایل ثانویه کدهایی که قرار است به اسمبلی برنامه اضافه شوند، تاثیر گذار بود.

**چند نکته تکمیلی در مورد توزیع برنامه‌های مبتنی بر PostSharp**

الف) اگر نیاز است به اسمبلی‌های خود امضای دیجیتال اضافه کنید، در حالت استفاده از PostSharp به علت بازنویسی کدهای IL اسمبلی تولیدی، نیاز است حالت delay signing انتخاب شود. به این معنا که ابتدا اسمبلی به صورت متداول کامپایل می‌شود. سپس PostSharp کار خود را انجام داده و در نهایت با استفاده از ابزارهای اعمال امضای دیجیتال باید کار افزودن آن‌ها در مرحله آخر انجام شود.
ب) در حال حاضر تنها برنامه Dotfuscator است که با PostSharp برای obfuscation سازگاری دارد.

**نظرات خوانندگان**

نویسنده: رضا1356
تاریخ: ۱۳۹۲/۱۰/۰۳ ۹:۴۵

اگر امکان دارد لینک مستقیمی جهت دریافت postsharp معرفی کنید.

ضمن اینکه با توجه به اینکه که فرمودید postsharp بسته تجاری و مدت دار است آیا درست است که پروژه امان را وابسته به آن کنیم.

تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۳ ۱۱:۵۱

- لینک مستقیمی ندارم. جهت تست از بسته NuGet آن استفاده کنید.
- SQL Server هم تجاری است. Windows هم از بنیان تجاری است. احتمالا از هر دوی این‌ها استفاده می‌کنید. تجاری بودن دلیلی برای سرکوب اشتیاق به یادگیری مطلبی نیست و نخواهد بود.
- ضمن اینکه در قسمت‌های بعدی نمونه‌های سورس باز هم معرفی شده‌اند.

نویسنده: رضا شش
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱۵:۵۳

با سلام
من آخرین نسخه postsharp رو از سایت نیوگت دریافت کردم در یک برنامه ساده HelloWord استفاده کردم وقتی دیباگ می‌کنم وارد قسمت Aspect نمی‌شود با چند dll مختلف امتحان کردم و فقط یک ورژن 2 پیدا کردم که جواب داد آیا تنظیمات خاصی نیاز دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱۷:۱۰

جزئیات مراحل اتصال Aspects در قسمت قبل بررسی شدند. همچنین این کتابخانه صرفا DLL ایی نیست. یک سری مراحل post build را باید به VS.NET اضافه کند تا پس از کامپایل اولیه برنامه، کار تغییر اسمبلی را انجام دهد.

نویسنده: رضا شش
تاریخ: ۱۳۹۲/۱۰/۰۵ ۹:۳۵

من مثال ذکر شده helloword را در سایت قرار می‌دهم. اسمبلی postsharp استفاده شده فقط ورژنش فرق می‌کند ولی در یکی کار می‌کند و در دیگری خیر.

HelloWorld.rar

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۵ ۱۰:۹

روی سیستم من هیچکدام از مثال‌های شما کار نکردند. دلایل:
الف) همانطور که عرض شد، PostSharp فقط یک DLL نیست (IL Weaving به معنای دستکاری کدهای IL و اسمبلی نهایی است و

افزودن کدهایی در این میان). بسته نیوگت آن، یک سری مراحل Post Build را به فایل csproj اضافه می‌کند؛ برای مثال:

```
<Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
  <Import Project="..\packages\PostSharp.2.1.7.30\tools\PostSharp.targets"
    Condition="Exists('..\packages\PostSharp.2.1.7.30\tools\PostSharp.targets')" />
```

ب) حتما باید سیستم licensing آن توسط نیوگت نصب شود تا عملیات IL Weaving را انجام دهد.
ج) زمانیکه از طریق نیوگت نصب می‌شود، پوشه packages\PostSharp.2.1.7.30\tools آن کار اصلی IL Weaving را انجام می‌دهد و این پوشه بالای 10 مگابایت است.

نویسنده: رضا شش
تاریخ: ۱۳۹۲/۱۰/۰۷ ۹:۵۰

با عرض معذرت چند سوال دارم:
-1 اینطور که من متوجه شدم اگر بخواهیم در هر پروژه ای از postsharp استفاده کنیم حتما باید به اینترنت وصل باشیم و بسته چندین مگاباتی نیوگت آن را نصب کنیم. اگر اینطور است در شرکت‌ها و سازمان‌ها همه سیستم‌ها اجازه دسترسی به اینترنت را ندارند.
-2 در پروژه من بعد از کامپایل یک پیغام در قسمت output درج می‌شود که می‌گوید چند روز تا انقضای این بسته فرصت دارید. پس از انقضای مهلت مقرر چکار باید کرد چون بنا دارم از این امکان در پروژه ام استفاده کنم.
-3 در مثالهایی دریافتی از اینترنت یک فایل اجرایی وجود دارد به نام PostSharp.MSBuild.Samples.exe این فایل چه کاربردی دارد. چون در سیستم من اجرا نمی‌شود.
ExceptionHandling.zip

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۷ ۱۰:۱۱

- بحث در مورد AOP بدون ذکر نامی از PostSharp بی‌معنا بود. به همین جهت چند قسمتی به آن اختصاص داده شد. حداقل از لحاظ بحث مفهومی ارزشمند است.
- در سازمان‌ها امکان تشکیل یک مخزن نیوگت محلی وجود دارد. یعنی فقط کافی است یکی از سیستم‌ها تبدیل به مخزن شود و بقیه از آن استفاده کنند. اطلاعات بیشتر در اینجا
- پیشنهاد من استفاده از پروژه‌های سورس باز مشابهی است مانند Fody. یک نمونه از کاربرد آن‌را در ادامه این دوره بررسی کرده‌ایم: « معرفی پروژه NotifyPropertyWeaver ». امکانات زیادی دارد . یا اینکه اصلا از IL Weaving استفاده نکنید و از dynamic proxy مطرح شده مانند پروژه castle core که در قسمت‌های قبل بررسی شد، استفاده نمائید.
- post sharp زمانیکه از طریق نیوگت نصب می‌شود، خودش را در سیستم build ویژوال استودیو مرتبط با پروژه جاری ثبت می‌کند. پس از اینکه dll یا فایل exe شما توسط VS.NET تولید شد، به صورت خودکار کار post sharp آغاز شده و کدهای IL اضافی پیاده سازی کننده aspects مدنظر را به اسمبلی‌های برنامه اضافه می‌کند.