عنوان: **استفاده از IL Code Weaving برای تولید ویژگی‌های تکراری مورد نیاز در WCF**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۲ ۰:۱۲
آدرس: www.dotnettips.info
برچسب‌ها: Design patterns, AOP, C#, Architecture

با استفاده از IL Code Weaving علاوه بر مدیریت اعمال تکراری پراکنده در سراسر برنامه مانند ثبت وقایع، مدیریت استثناءها، کش کردن داده‌ها و غیره، می‌توان قابلیتی را به کدهای موجود نیز افزود. برای مثال یک برنامه معمول WCF را درنظر بگیرید.

```
using System.Runtime.Serialization;

namespace AOP03.DataContracts
{
    [DataContract]
    public class User
    {
        [DataMember]
        public int Id { set; get; }

        [DataMember]
        public string Name { set; get; }
    }
}
```

نیاز است کلاس‌ها و خواص آن توسط ویژگی‌های DataContract و DataMember مزین شوند. در این بین نیز اگر یکی فراموش گردد، کار دیباگ برنامه مشکل خواهد شد و در کل حجم بالایی از کدهای تکراری در اینجا باید در مورد تمام کلاس‌های مورد نیاز انجام شود. در ادامه قصد داریم تولید این ویژگی‌ها را توسط PostSharp انجام دهیم. به عبارتی یک پوشه خاص به نام DataContracts را ایجاد کرده و کلاس‌های خود را به نحوی متداول و بدون اعمال ویژگی خاصی تعریف کنیم. در ادامه پس از کامپایل آن، به صورت خودکار با ویرایش کدهای IL توسط PostSharp، ویژگی‌های لازم را به اسمبلی نهایی اضافه نمائیم.

تهیه DataContractAspect جهت اعمال خودکار ویژگی‌های DataContract و DataMember

```
using System;
using System.Collections.Generic;
using System.Reflection;
using System.Runtime.Serialization;
using PostSharp.Aspects;
using PostSharp.Extensibility;
using PostSharp.Reflection;

namespace AOP03
{
    [Serializable]
    //این ویژگی تنها نیاز است به کلاس‌ها اعمال شود
    [MulticastAttributeUsage(MulticastTargets.Class)]
    public class DataContractAspect : TypeLevelAspect, IAspectProvider
    {
        public IEnumerable<AspectInstance> ProvideAspects(object targetElement)
        {
            var targetType = (Type)targetElement; //همان نوعی است که ویژگی جاری به آن اعمال خواهد شد

            //این سطر معادل است با درخواست تولید ویژگی دیتاکانترکت
            var introduceDataContractAspect = new CustomAttributeIntroductionAspect(
                new ObjectConstruction(typeof(DataContractAttribute).GetConstructor(Type.EmptyTypes)));

            //این سطر معادل است با درخواست تولید ویژگی دیتاممبر
            var introduceDataMemberAspect = new CustomAttributeIntroductionAspect(
                new ObjectConstruction(typeof(DataMemberAttribute).GetConstructor(Type.EmptyTypes)));

            //در اینجا کار اعمال ویژگی دیتاکانترکت به کلاسی که به عنوان پارامتر متد جاری
            //دریافت شده انجام خواهد شد
            yield return new AspectInstance(targetType, introduceDataContractAspect);

            //مرحله بعد کار اعمال ویژگی دیتاممبر به خواص کلاس است
            foreach (var property in targetType.GetProperties(BindingFlags.Public |
                                                               BindingFlags.DeclaredOnly |
                                                               BindingFlags.Instance))
            {
```

```
                if (property.CanWrite)
                    yield return new AspectInstance(property, introduceDataMemberAspect);
            }
        }
    }
}
```

توضیحات مرتبط با قسمت‌های مختلف این Aspect سفارشی، به صورت کامنت در کدهای فوق ارائه شده‌اند.
برای اعمال آن به سراسر برنامه تنها کافی است به فایل AssemblyInfo.cs پروژه مراجعه و سپس سطر زیر را به آن اضافه کنیم:

```
[assembly: DataContractAspect(AttributeTargetTypes = "AOP03.DataContracts.*")]
```

به این ترتیب در زمان کامپایل پروژه، Aspect تعریف شده به تمام کلاس‌های موجود در فضای نام AOP03.DataContracts اعمال خواهند شد.

در این حالت اگر کلیه ویژگی‌های کلاس User فوق را حذف و برنامه را کامپایل کنیم، با مراجعه به برنامه ILSpy می‌توان صحت اعمال ویژگی‌ها را به کمک PostSharp بررسی کرد:

```
using System;
using System.Diagnostics;
using System.Runtime.CompilerServices;
using System.Runtime.Serialization;
namespace AOP03.DataContracts
{
    [DataContract]
    public class User
    {
        [System.Runtime.CompilerServices.CompilerGenerated, System.Diagnostics.DebuggerNonUserCode]
        internal sealed class <>z__Aspects...
        [DataMember]
        public int Id...
        [DataMember]
        public string Name...
    }
}
```