

هنگامی که از روش [AOP](#) استفاده می‌کنیم گاهی نیاز است متد تزئین‌شده را از متدی درون خود کلاس فراخوانی کنیم و می‌خواهیم aspectهای آن متد نیز فراخوانی شوند.

پیش‌نیاز: دوره‌ی AOP

(برای سادگی کار از تعریف attribute خودداری کردم. شما می‌توانید با توجه به آموزش، attributeهای دلخواه را به متدها بیافزایید).

Interface و کلاس پیاده‌سازی‌شده‌ی آن در لایه سرویس:

```
public interface IMyService
{
    void foo();
    void bar();
}

public class MyService : IMyService
{
    public void foo()
    {
        Console.WriteLine("foo");
        bar();
    }

    public void bar()
    {
        Console.WriteLine("bar");
    }
}
```

نام متد در خروجی نوشته می‌شود. همچنین می‌خواهیم پیش از فراخوانی این متدها، متنی در خروجی نوشته شود.

آماده‌سازی `Interceptor`

یک `interceptor` ساده که نام متد را در خروجی می‌نویسد.

```
//using Castle.DynamicProxy;

public class Interceptor : IInterceptor
{
    public void Intercept(IInvocation invocation)
    {
        Console.WriteLine("Intercepted: " + invocation.Method.Name);
        invocation.Proceed();
    }
}
```

معرفی `Interceptor` به سیستم

همانند قبل:

```
//using System;
//using Castle.DynamicProxy;
//using StructureMap;

class Program
{
    static void Main(string[] args)
    {
        ObjectFactory.Initialize(x =>
        {
            var dynamicProxy = new ProxyGenerator();
            x.For<IMyService>()
                .EnrichAllWith(myTypeInterface =>
                    dynamicProxy.CreateInterfaceProxyWithTarget(myTypeInterface, new
Intercept()))
                .Use<MyService>();
        });
    }
}
```

```

    });
    var myService = ObjectFactory.GetInstance<IMyService>();
    myService.foo();
}
}

```

انتظار ما این است که خروجی زیر تولید شود:

```

Intercepted foo
foo
Intercepted bar
bar

```

اما نتیجه این می‌شود که دلخواه ما نیست:

```

Intercepted foo
foo
bar

```

راه حل

برای حل این مشکل دو کار باید انجام داد:
1- متد تزئین‌شده باید **virtual** باشد.

```

public class MyService : IMyService
{
    public virtual void foo()
    {
        Console.WriteLine("foo");
        bar();
    }

    public virtual void bar()
    {
        Console.WriteLine("foo");
        bar();
    }
}

```

2- شیوه معرفی متد به سیستم باید به روش زیر باشد:

```

// جایگزین روش پیشین در متد
x.For<IMyService>()
    .EnrichAllWith(myTypeInterface => dynamicProxy.CreateClassProxy<MyService>(new
Intercept()))
    .Use<MyService>();

```

دلیل این مسئله به دو روش proxy برمی‌گردد که برای اطلاع بیشتر به [مستندات پروژه Castle](#) مراجعه کنید.
در این‌جا روش Inheritance-based به کار رفته است. در این روش، تنها متدهای **virtual** را می‌توان intercept کرد. در روش پیشین (Composition-based) برای تمامی متدها عملیات intercept انجام می‌شد (کلاس proxy پیاده‌سازی‌شده‌ی interface ما بود) که در این‌جا این‌گونه نیست و می‌تواند به سرعت برنامه کمک کند.