

بسیار پیش می‌آید که یک کنترلر را به یک [اکشن فیلتر](#) خاص مزین کنیم. در این صورت تمامی اکشن‌های موجود در کنترلر مربوطه مجاب به اجرای [اکشن فیلتر](#) مورد نظر می‌شوند. اما بسیار پیش می‌آید که نخواهیم یک اکشن خاص در کنترلر مذکور [اکشن فیلتر](#) مورد نظر را اجرا کند.

یک راهکار ساده اما (به نظر شخصی من) غیر منطقی این است که تک تک اکشن‌هایی را که می‌خواهیم [اکشن فیلتر](#) مورد نظر را اجرا کنند، مزین کنیم و اکشن‌هایی که نمی‌خواهیم [اکشن فیلتر](#) مورد نظر را اجرا کنند به [اکشن فیلتر](#) مورد نظر مزین نمی‌کنیم. اما فرض کنید تعداد اکشن‌های ما زیاد باشند؛ به نظر این روش غیر منطقی و غیر بهینه است.

یکی از مشکلاتی که در یکی از پروژه‌ها حدود سه روز وقت من را گرفت همین کار بود. زمانی که از یک [تصویر امنیتی](#) جهت مقابله با ربات‌های استفاده می‌کردم به این مشکل برخورد کردم. کنترلر من به یک [اکشن فیلتر](#) فشرده سازی محتوا مزین بود. در نتیجه اکشنی که تصویر امنیتی را تولید میکرد نیز [اکشن فیلتر](#) فشرده سازی را اجرا می‌کرد و پس از فشرده سازی باعث می‌شد که تصویر امنیتی نشان داده نشود، چون فرمت تصویری آن بهم ریخته بود. یک راهکار را که پس از جستجو به آن رسیدم، جهت استفاده‌ی دوستان مطرح می‌کنم.

برای اینکه از اجرای چنین [اکشن فیلترهایی](#) جلوگیری کنیم نیاز است کمی [اکشن فیلتر](#) مورد نظر را دستکاری کنیم.

برای این کار باید مراحل زیر را انجام داد:

1- ابتدا یک [Attribute](#) خالی را تعریف می‌کنیم.

2- سپس [اکشن فیلتر](#) دلخواهی را تعریف کرده و در زمان اجرا بررسی می‌کنیم اگر متد (اکشن) مورد نظر با [Attribute](#) تعریفی در مرحله یک مزین شده بود، در نتیجه [اکشن فیلتر](#) را اجرا نمی‌کنیم.

3- هر اکشنی را که نمی‌خواهیم [اکشن فیلتر](#) تعریفی مرحله 2 را اجرا کند، آن را به [Attribute](#) مرحله یک مزین می‌کنیم.

به این ترتیب می‌توانیم از اجرای [اکشن فیلتر](#) دلخواه روی متدها یا اکشن‌های دلخواه جلوگیری کنیم. در ادامه نحوه‌ی تعریف آنها را در زیر مشاهده می‌کنید.

1- تعریف یک [Attribute](#) دلخواه مثلاً با نام DisableCompression

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = false, Inherited = true)]
public sealed class DisableCompression : Attribute { }
```

2- تعریف [اکشن فیلتر](#) دلخواه مثلاً با نام CompressionFilter

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, Inherited = true, AllowMultiple = false)]
public class CompressionFilter : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        bool disabled = filterContext.ActionDescriptor.IsDefined(typeof(DisableCompression), true) ||
        filterContext.ActionDescriptor.ControllerDescriptor.IsDefined(typeof(DisableCompression), true);
        if (disabled)
            return;

        // کدهای دلخواه اکشن فیلتر مورد نظر
    }
}
```

3- در این مرحله هر اکشنی را که نمی‌خواهیم [اکشن فیلتر](#) CompressionFilter را اجرا کند به [Attribute](#) با نام

DisableCompression مزین میکنیم.

```
[CompressionFilter]
public abstract class BaseController : Controller
{
}

public class SomeController : BaseController
{
    public ActionResult WantThisActionCompressed()
    {
        // code
    }

    [DisableCompression]
    public ActionResult DontWantThisActionCompressed()
    {
        // code
    }
}
```

با این کار اکشن `WantThisActionCompressed` [اکشن فیلتر](#) `CompressionFilter` را اجرا می‌کند اما اکشن `DontWantThisActionCompressed` چون مزین به `DisableCompression` شده‌است، پس در نتیجه [اکشن فیلتر](#) `CompressionFilter` بر روی آن اجرا نخواهد شد.