عنوان: **تزریق وابستگی‌ها در فیلترهای ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۱۸ ۱:۷
آدرس: www.dotnettips.info
گروه‌ها: Design patterns, Dependency Injection, IoC

فرض کنید فیلتر سفارشی لاگ کردن را که از سرویس ILogActionService استفاده می‌کند، به نحو ذیل تعریف کرده‌اید:

```
public interface ILogActionService
{
    void Log(string data);
}

public class LogAttribute : ActionFilterAttribute
{
    public ILogActionService LogActionService { get; set; }

    public override void OnActionExecuted(ActionExecutedContext filterContext)
    {
        LogActionService.Log("......data......");
        base.OnActionExecuted(filterContext);
    }
}
```

با استفاده‌ای مانند:

```
[Log]
public ActionResult Index()
{}
```

روش متداول تنظیمات تزریق وابستگی‌ها در ASP.NET MVC، بیشتر به بحث کنترلرها مرتبط است و سایر قسمت‌ها را پوشش نمی‌دهد. برای این مورد خاص ابتدا نیاز است یک FilterProvider سفارشی را به نحو ذیل تدارک دید:

```
using StructureMap;
using System.Collections.Generic;
using System.Web.Mvc;

namespace DI06.CustomFilters
{
    public class StructureMapFilterProvider : FilterAttributeFilterProvider
    {
        private readonly IContainer _container;
        public StructureMapFilterProvider(IContainer container)
        {
            _container = container;
        }

        public override IEnumerable<Filter> GetFilters(ControllerContext controllerContext,
ActionDescriptor actionDescriptor)
        {
            var filters = base.GetFilters(controllerContext, actionDescriptor);
            foreach (var filter in filters)
            {
                _container.BuildUp(filter.Instance);
                yield return filter;
            }
        }
    }
}
```

نکته‌ی مهم آن، استفاده از متد BuildUp استراکچرمپ است. نمونه‌ی آن‌را در تنظیمات تزریق وابستگی‌ها در وب فرم‌ها پیشتر ملاحظه کرده‌اید . در این مثال کار آن وهله سازی وابستگی‌های فیلترهای تعریف شده در برنامه است.
پس از اینکه FilterProvider سفارشی مخصوص کار با استراکچرمپ را تهیه کردیم، اکنون نوبت به جایگزین کردن آن با FilterProvider پیش فرض ASP.NET MVC در فایل global.asax.cs به نحو ذیل است:

```
//Using the custom StructureMapFilterProvider
var filterProvider = FilterProviders.Providers.Single(provider => provider is
```

```
FilterAttributeFilterProvider);
FilterProviders.Providers.Remove(filterProvider);
FilterProviders.Providers.Add(SmObjectFactory.Container.GetInstance<StructureMapFilterProvider>());
```

استفاده از SmObjectFactory.Container.GetInstance سبب خواهد شد تا به صورت خودکار، وابستگی تزریق شده‌ی در سازنده‌ی کلاس StructureMapFilterProvider وهله سازی و تامین شود.

همچنین در این مثال چون تزریق وابستگی در کلاس LogAttribute از نوع setter injection است، نیاز است در تنظیمات ابتدایی Container مورد استفاده، Policies.SetAllProperties نیز قید شود:

```
namespace DI06.IocConfig
{
    public static class SmObjectFactory
    {
        private static readonly Lazy<Container> _containerBuilder =
            new Lazy<Container>(defaultContainer, LazyThreadSafetyMode.ExecutionAndPublication);

        public static IContainer Container
        {
            get { return _containerBuilder.Value; }
        }

        private static Container defaultContainer()
        {
            return new Container(x =>
            {
                x.For<ILogActionService>().Use<LogActionService>();

                x.Policies.SetAllProperties(y =>
                {
                    y.OfType<ILogActionService>();
                });
            });
        }
    }
}
```

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

DI06

نظرات خوانندگان

نویسنده: محسن موسوی
تاریخ: ۱۳۹۴/۰۶/۱۴ ۱۲:۳۵

با تشکر
در مورد GlobalFilter راه حلی هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۶/۱۴ ۱۲:۴۱

به چه مشکلی برخوردید؟

نویسنده: محسن موسوی
تاریخ: ۱۳۹۴/۰۶/۱۴ ۱۳:۲

راه حل کنونی globalfilters پوشش نمیده.
البته دنبال راه حل جامع و شفاف مثل مقاله جاری هستم.
یکی از راه‌های پیشنهادی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۶/۱۴ ۱۴:۵۳

در مورد فیلترهای سراسری، حلقه‌ی زیر در کلاس StructureMapFilterProvider فراخوانی نخواهد شد:

```
var filters = base.GetFilters(controllerContext, actionDescriptor);
foreach (var filter in filters)
```

بنابراین container.BuildUp ایی هم بر روی فیلتر در حال اجرا، فراخوانی نمی‌شود و وابستگی‌های آن تامین نخواهند شد.
برای حل این مشکل، بجای روش معمول معرفی فیلترهای سراسری:

```
GlobalFilters.Filters.Add(new LogAttribute());
```

بنویسید:

```
GlobalFilters.Filters.Add(SmObjectFactory.Container.GetInstance<LogAttribute>());
```

در این حالت، در ابتدای کار برنامه، تمام وابستگی‌های مرتبط با LogAttribute هم وهله سازی می‌شوند.

مشکل!
این وهله سازی، فقط یکبار آن هم در ابتدای برنامه انجام می‌شود. یعنی وابستگی‌های استفاده شده‌ی در فیلتر سراسری، صرفنظر از طول عمر تعریف شده‌ی برای آن‌ها توسط IoC Container، دیگر وهله سازی مجدد نخواهند شد و این مساله برای حالت‌هایی مانند کار با دیتابیس مشکل ساز است.
برای حل این مشکل، اینترفیس IContainer را به فیلتر تزریق کنید:

```
public class LogAttribute : ActionFilterAttribute
{
    private readonly IContainer _container;

    //نباید به این صورت تعریف شود چون در فیلترهای سراسری فقط یکبار وهله سازی خواهد شد
    //public ILogActionService LogActionService { get; set; }
```

```
    public LogAttribute(IContainer container)
    {
        _container = container;
    }

    public override void OnActionExecuted(ActionExecutedContext filterContext)
    {
        _container.GetInstance<ILogActionService>().Log("......data......");
        //LogActionService.Log("......data......");
        base.OnActionExecuted(filterContext);
    }
}
```

در این حالت هرچند کلاس LogAttribute ایی که به صورت فیلتر سراسری تعریف شده‌است، یکبار در آغاز کار برنامه وهله سازی می‌شود، اما وابستگی‌های مورد نیاز آن، توسط container.GetInstance به ازای هر بار فراخوانی، مجددا ساخته خواهند شد و دیگر تک وهله‌ای نخواهند بود.

نویسنده: محسن موسوی
تاریخ: ۱۳۹۴/۰۶/۱۴ ۱۶:۲۲

با تشکر از زحمات فراوان شما، راه حل دیگر :

```
public class StructureMapGlobalFilterProvider : IFilterProvider
    {
        public StructureMapGlobalFilterProvider(IContainer container, GlobalFilterRegistrationList
filterList)
        {
            _container = container;
            _filterList = filterList;
        }

        private IContainer _container;
        private GlobalFilterRegistrationList _filterList;

        public IEnumerable<Filter> GetFilters(ControllerContext controllerContext, ActionDescriptor
actionDescriptor)
        {
            var filters = new List<Filter>();
            if (_filterList == null || _filterList.Count == 0)
                return filters;
            foreach (GlobalFilterRegistration registration in _filterList)
            {
                var actionFilter = _container.GetInstance(registration.Type);
                var filter = new Filter(actionFilter, FilterScope.Global, registration.Order);
                filters.Add(filter);
            }
            return filters;
        }
    }

    public class GlobalFilterRegistration
    {
        public Type Type { get; set; }
        public int? Order { get; set; }
    }

    public class GlobalFilterRegistrationList : List<GlobalFilterRegistration>
    {
    }
```

و تنظیمات Global:

```
var globalFilterRegistrationList = new GlobalFilterRegistrationList
                {
                    new GlobalFilterRegistration
                    {
                        Type = typeof (LogAttribute),
                        Order
                            = 1
                    }
                };
```

```
container.Configure(x =>
{
    x.For<IFilterProvider>().Use<StructureMapGlobalFilterProvider>();
    x.For<GlobalFilterRegistrationList>().Use(globalFilterRegistrationList);
});
```