عنوان: **تزریق وابستگی‌های AutoMapper در لایه سرویس برنامه**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۸ ۱۰:۱۹
آدرس: www.dotnettips.info
گروه‌ها: MVC, Entity framework, AutoMapper

اگر مطلب « Refactoring به تزریق وابستگی‌ها » را به خاطر داشته باشید، جهت تشخیص وابستگی‌های یک کلاس، کار از بررسی کلمات new و همچنین فراخوانی‌های استاتیک، شروع می‌شود و ... متد استاتیک Mapper.Map کتابخانه‌ی AutoMapper نیز از همین دست است. در ادامه قصد داریم بجای فراخوانی مستقیم Mapper.Map از اینترفیس IMappingEngine به عنوان تامین کننده‌ی متد Map استفاده کنیم. همچنین کلاس‌های Profile نوشته شده را نیز به صورت خودکار به برنامه اضافه نمائیم.

تنظیمات IoC Container مختص به AutoMapper

در ذیل یک کلاس Registry مخصوص StructureMap را مشاهده می‌کنید که جهت کپسوله کردن اطلاعات خاص AutoMapper تهیه شده‌است. می‌توان این اطلاعات را در داخل تنظیمات new Container خود قرار داد و یا می‌توان آن‌ها را جهت شلوغ نشدن سایر تنظیمات IoC Container، به یک کلاس Registry منتقل کرد:

```
public class AutomapperRegistry : Registry
{
    public AutomapperRegistry()
    {
        var platformSpecificRegistry = PlatformAdapter.Resolve<IPlatformSpecificMapperRegistry>();
        platformSpecificRegistry.Initialize();

        For<ConfigurationStore>().Singleton().Use<ConfigurationStore>()
            .Ctor<IEnumerable<IObjectMapper>>().Is(MapperRegistry.Mappers);

        For<IConfigurationProvider>().Use(ctx => ctx.GetInstance<ConfigurationStore>());

        For<IConfiguration>().Use(ctx => ctx.GetInstance<ConfigurationStore>());

        For<ITypeMapFactory>().Use<TypeMapFactory>();

        For<IMappingEngine>().Singleton().Use<MappingEngine>()
                             .SelectConstructor(() => new MappingEngine(null));

        this.Scan(scanner =>
        {
            scanner.AssembliesFromApplicationBaseDirectory();

            scanner.ConnectImplementationsToTypesClosing(typeof(ITypeConverter<,>))
                   .OnAddedPluginTypes(t => t.HybridHttpOrThreadLocalScoped());

            scanner.ConnectImplementationsToTypesClosing(typeof(ValueResolver<,>))
                .OnAddedPluginTypes(t => t.HybridHttpOrThreadLocalScoped());
        });
    }
}
```

هدف اصلی، وهله سازی خودکار IMappingEngine است و برای رسیدن به آن، باید تمام وابستگی‌های کلاس MappingEngine را مانند IConfigurationProvider و سایر مواردی که مشاهده می‌کنید، مشخص کرد. پس از این تنظیمات، کلاس ObjectFactory سفارشی برنامه به شکل ذیل جهت معرفی AutomapperRegistry تغییر خواهد کرد:

```
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()
    {
        var container = new Container(cfg =>
        {
```

```
            cfg.AddRegistry<AutomapperRegistry>();
            cfg.Scan(scan =>
            {
                scan.TheCallingAssembly();
                scan.WithDefaultConventions();
                scan.AddAllTypesOf<Profile>().NameBy(item => item.FullName);
            });
        });

        configureAutoMapper(container);

        return container;
    }

    private static void configureAutoMapper(IContainer container)
    {
        var configuration = container.TryGetInstance<IConfiguration>();
        if (configuration == null) return;
        //saying AutoMapper how to resolve services
        configuration.ConstructServicesUsing(container.GetInstance);
        foreach (var profile in container.GetAllInstances<Profile>())
        {
            configuration.AddProfile(profile);
        }
    }
}
```

در اینجا علاوه بر معرفی AutomapperRegistry، یک مورد دیگر نیز اضافه شده‌است: یافتن خودکار کلاس‌هایی از نوع Profile و همچنین فراخوانی متد AddProfile کتابخانه‌ی AutoMapper به صورت خودکار. به این ترتیب دیگر نیازی نخواهد بود تا در ابتدای کار برنامه، متد Mapper.Initialize را جهت معرفی کلاس‌های Profile فراخوانی کرد و این‌کار به صورت خودکار توسط متد configureAutoMapper انجام می‌شود.

تغییرات لایه سرویس برنامه جهت استفاده از IoC Container

اکنون که IoC Container ما با نحوه‌ی یافتن وابستگی‌های IMappingEngine آشنا شده‌است، تنها کافی است این اینترفیس را در سازنده‌ی کلاس سرویس خود تزریق کنیم:

```
public class UsersService : IUsersService
{
    private readonly IMappingEngine _mappingEngine;

    public UsersService(IMappingEngine mappingEngine)
    {
        _mappingEngine = mappingEngine;
    }

    public UserViewModel GetName(int id)
    {
        var dbUser1 = new User
        {
            Id = 1,
            Name = "Test",
            RegistrationDate = DateTime.Now.AddDays(-10)
        };

        var uiUser = new UserViewModel();
        _mappingEngine.Map(source: dbUser1, destination: uiUser);
        return uiUser;
    }
}
```

و پس از آن از متد Map این اینترفیس بجای فراخوانی مستقیم Mapper.Map می‌توان استفاده کرد. به این ترتیب وابستگی مورد نیاز این کلاس، از طریق سازنده‌ی آن به آن تزریق شده‌است و دیگر فراخوانی‌های استاتیک را در اینجا مشاهده نمی‌کنیم.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

AM_Sample03.zip

نظرات خوانندگان

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۲/۰۹ ۱۱:۰

ممنون از شما،
یک سوال: بنده کلاس ObjectFactory را همانطور که فرمودید به این صورت تغییر دادم. در لایه سرویس نیز این متد را تهیه کرده‌ام:

```
public IList<AdvertismentViewModel> GetAdvertisementsByMe(int userId)
{
            var adsList = _advertisements.Where(x => x.UserId == userId).ToList();
            var adsViewModel = new List<AdvertismentViewModel>();
            _mappingEngine.Map(source: adsList, destination: adsViewModel);
            return adsViewModel;
}
```

در متد فوق کلاس Advertisment به کلاس زیر نگاشت داده شده است:

```
public class AdvertismentViewModel
    {
        public string Image { get; set; }
        public string Title { get; set; }
        public string ExpireDate { get; set; }
    }
```

اما با فراخوانی متد GetAdvertisementsByMe استثناء AutoMapperMappingException صادر می‌شود:

```
Missing type map configuration or unsupported mapping.

Mapping types:
Advertisement -> AdvertismentViewModel
Project.DomainClasses.Advertisement -> Project.Models.AdvertismentViewModel

Destination path:
List`1[0]

Source value:
System.Data.Entity.DynamicProxies.Advertisement_E82DFF273E08C95AA785F8F7A0D2B5ABC8E54C4566DFE1C8A92D8D3
C447608AE
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۹ ۱۱:۳۱

- آیا کلاس پروفایل این نگاشت مورد نظر تعریف شده‌است (کلاس حاوی CreateMap)؟
- اگر بله، در متد configureAutoMapper روی سطر configuration.AddProfile یک break point قرار دهید و بررسی کنید که آیا فراخوانی می‌شود؟ یعنی آیا به صورت خودکار یافت شده و به سیستم اضافه می‌شود یا خیر؟
- اگر این break point فراخوانی نمی‌شود، این کلاس پروفایل در چه اسمبلی قرار دارد؟ بازه‌ی اسکن استراکچرمپ را باید تغییر دهید یا وسیع‌تر کنید. برای مثال scan.TheCallingAssembly فقط اسمبلی فراخوان را اسکن می‌کند. اگر نیاز است اسمبلی دیگری هم اسکن شود، از متد AssemblyContainingType استفاده کنید:

```
Scan(scan =>
    {
        scan.AssemblyContainingType<یکی از کلاس‌های پروفایل این اسمبلی خاص>();
        //....
    });
```

نویسنده: وحید نصیری

تاریخ: ۱۳۹۴/۰۲/۰۹ ۱۱:۴۲

یک نکته‌ی تکمیلی:

اگر با EF کار می‌کنید و LINQ to Objects نیست، از متد Project To بهتر است استفاده کنید:

```
ctx.Advertisements.Where(...).Project().To<AdvertisementViewModel>().ToList()
```

مزیت این روش این است که فقط خواص موجود در ViewModel از بانک اطلاعاتی واکشی می‌شوند؛ برای نمونه در اینجا و این مثال، فقط سه مورد.

اگر ابتدا ToList خود EF را فراخوانی کنید، تمام خواص کلاس Advertisment از بانک اطلاعاتی واکشی خواهند شد و مرحله‌ی بعد LINQ to Objects می‌شود.

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۲/۰۹ ۱۱:۴۶

:) خیلی ممنون، من اصلاً حواسم نبود دقیقاً مشکل همین بود.

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۲/۰۹ ۱۳:۴۵

در این صورت نگاشت کلاس‌ها باید داخل لایه سرویس توسط Mapper.Map صورت گیرد:

```
public IList<AdvertismentViewModel> GetAdvertisementsByMe(int userId)
        {
            Mapper.CreateMap<Advertisement, AdvertismentViewModel>();
            var adsList = _advertisements.Where(x => x.UserId ==
userId).Project().To<AdvertismentViewModel>().ToList();
            return adsList;
        }
```

چون در غیر اینصورت استثنای InvalidOperationException صادر خواهد شد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۹ ۱۴:۱۹

کاری که در اینجا انجام شده، ایجاد یک Mapping Engine سفارشی هست که با Mapping Engine اصلی استاتیک یکی نیست. به همین جهت برای نمونه متد Project آرگومان (**_mappingEngine**)Project هم دارد. اگر قید نشود، یعنی قرار است از موتور نگاشت استاتیک سراسری پیش فرض آن استفاده شود.

نویسنده: مجتبی آزاد
تاریخ: ۱۳۹۴/۰۳/۰۶ ۱۴:۲۰

سلام؛ محل صحیح قرار دادن Mappingها دقیقا کجای پروژه است؟ آیا مثال همین مطلب صحیح‌ترین محل قرار دادن Mappingها و AutoMapper است؟

در پروژه‌های مختلف و بعضی از مطالب همین وبسایت دیده‌ام که محل دیگری غیر از پروژه Service نیز برای قرار دادن Mappingها انتخاب می‌شود.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۰۶ ۱۴:۳۴

- «... صحیح‌ترین ...» «... محل دیگری غیر از پروژهService ...»

این «مثال» اساسا یک پروژه بیشتر نبود؛ صرفا جهت نمایش مفهوم مورد بحث. در همین «مثال» تعاریف نگاشت‌ها داخل پوشه‌ی سرویس نیست .

در کل می‌توانید یک اسمبلی جداگانه برای آن درنظر بگیرید به نام مثلا AutoMapperConfig. تنها قسمت مهم آن، بارگذاری و خواندن این نگاشت‌ها در زمان آغاز برنامه است که در مثال جاری، اینکار توسط SmObjectFactory به صورت خودکار انجام می‌شود.

در کل هدف از اکثر مثال‌های این سایت یا سایت‌های مشابه دیگر، رساندن یک مفهوم است؛ نه ارائه‌ی یک راه حل جامع و مانع. همینقدر که مثال زده شده، عنوان مورد بحث را پوشش دهد، کافی است.

نویسنده: مجتبی آزاد
تاریخ: ۱۳۹۴/۰۳/۰۶ ۱۵:۴۲

ممنونم از پاسختون.
هدف من بیشتر از طرح این سوال این هست که در طراحی معماری پروژه و به طور خاص جایگاه Mapping در پروژه، بین دو مورد تصمیم گیری کنم:
۱- قرار دادن تعاریف Mapping و model viewها در لایه UI و استفاده از لایه سرویس (با خروجی Entity Model در هر تابع)
۲- قراردادن تعاریف Mapping و model viewها هر کدام در یک پروژه مجزا و استفاده از آن در لایه سرویس، با این توضیح که خروجی متدها در لایه سرویس Viewmodel باشد
کدام یک از این موارد صحیح‌تر هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۰۶ ۱۶:۱

- محل **تعریف** نگاشت‌ها و کلاس‌های پروفایل، مهم نیست. چون اساسا هرجایی که قرار گیرند، دو وابستگی بیشتر نخواهند داشت: کلاس‌های مدل و کلاس‌های ViewModel.
- محل فراخوانی اولیه‌ی تعاریف نگاشت‌ها جهت معرفی آن‌ها به سیستم، مهم است.
+ اگر از کاربر اطلاعاتی را دریافت می‌کنید، در لایه UI هست که کار نگاشت اطلاعات دریافتی از کاربر و از ViewModelها به Modelهای اصلی برنامه انجام می‌شود (توسط متد Mapper.Map). اگر قرار است اطلاعاتی را بازگشت دهید، متدهای جدیدی مانند Project To بسیار بهینه‌تر هستند از روش قدیمی Mapper.Map و این متد را بهتر است در لایه سرویس استفاده کنید. متد Project To کارش بهینه سازی کوئری SQL ارسالی به سرور هست. اگر از روش Mapper.Map در لایه UI استفاده کنید، این قابلیت را از دست خواهید داد؛ چون Mapper.Map به معنای کار با اشیاء درون حافظه و LINQ to Objects است. کار متد ویژه‌ی Project To افزونه‌ای برای کار با Entity Framework و بهینه سازی آن است.

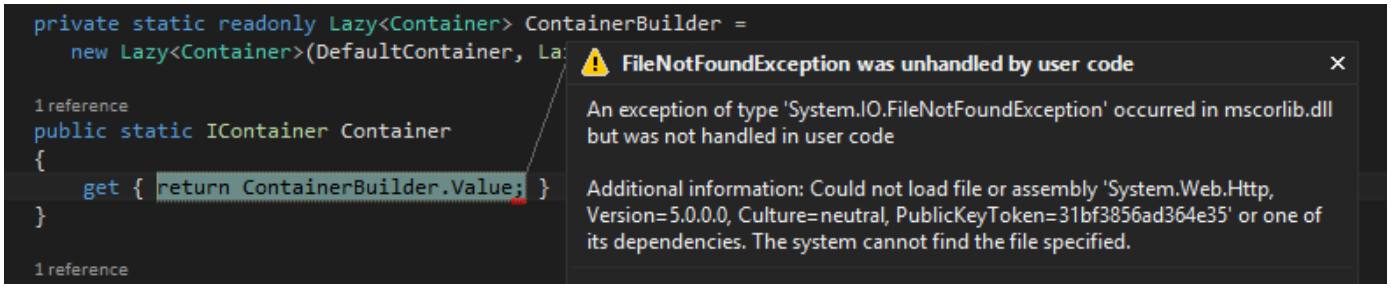
نویسنده: مجتبی آزاد
تاریخ: ۱۳۹۴/۰۳/۰۹ ۱۳:۲

من تنظیمات تزریق وابستگی مربوط به AutoMapper را در همان محل قرارگیری تزریق وابستگی Serviceها قرار داده ام و ObjectFactory به این شکل شد:

```
public static class WoObjectFactory
    {
        private static readonly Lazy<Container> ContainerBuilder =
           new Lazy<Container>(DefaultContainer, LazyThreadSafetyMode.ExecutionAndPublication);

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

        private static Container DefaultContainer()
        {

            var _container = new Container(x =>
            {
                var platformSpecificRegistry =
PlatformAdapter.Resolve<IPlatformSpecificMapperRegistry>();
                platformSpecificRegistry.Initialize();
```

```
x.For<ConfigurationStore>().Singleton().Use<ConfigurationStore>().Ctor<IEnumerable<IObjectMapper>>().Is
(MapperRegistry.Mappers);
                x.For<IConfigurationProvider>().Use(ctx => ctx.GetInstance<ConfigurationStore>());
                x.For<IConfiguration>().Use(ctx => ctx.GetInstance<ConfigurationStore>());
                x.For<ITypeMapFactory>().Use<TypeMapFactory>();
                x.For<IMappingEngine>().Singleton().Use<MappingEngine>().SelectConstructor(() => new
MappingEngine(null));

                x.For<IUnitOfWork>().HybridHttpOrThreadLocalScoped().Use(() => new
WirelessOrganizationContext());

                x.Scan(scan =>
                {
                    scan.AssemblyContainingType<IDeviceService>();
                    scan.TheCallingAssembly();
                    scan.WithDefaultConventions();
                });
                x.Scan(scanner =>
                {
                    scanner.AssembliesFromApplicationBaseDirectory();

                    scanner.ConnectImplementationsToTypesClosing(typeof(ITypeConverter<,>))
                           .OnAddedPluginTypes(t => t.HybridHttpOrThreadLocalScoped());

                    scanner.ConnectImplementationsToTypesClosing(typeof(ValueResolver<,>))
                        .OnAddedPluginTypes(t => t.HybridHttpOrThreadLocalScoped());
                });

                x.Scan(scan =>
                {
                    scan.TheCallingAssembly();
                    scan.WithDefaultConventions();
                    scan.AddAllTypesOf<Profile>().NameBy(item => item.FullName);
                });


            });
            ConfigureAutoMapper(_container);
            return _container;
        }

        private static void ConfigureAutoMapper(IContainer container)
        {
            var configuration = container.TryGetInstance<IConfiguration>();
            if (configuration == null) return;
            //saying AutoMapper how to resolve services
            configuration.ConstructServicesUsing(container.GetInstance);
            foreach (var profile in container.GetAllInstances<Profile>())
            {
                configuration.AddProfile(profile);
            }
        }
    }
```

اما وقتی پروژه اجرا می‌شود اکسپشن زیر اتفاق می‌افتد، که البته بعد از اضافه کردن تنظیمات مربوط به تزریق وابستگی AutoMapper اتفاق افتاد، در این Value مربوط به ContainerBuilder مقداری ندارد:



نویسنده: وحید نصیری

تاریخ: ۱۳۹۴/۰۳/۰۹ ۱۳:۳۳

- استثنای صادر شده مربوط است به یافت نشدن اسمبلی System.Web.Http. در لیست ارجاعات برنامه، این ارجاع را یافته و خاصیت copy to local آن‌را true کنید؛ چیزی شبیه به این مشکل
- همچنین اگر Solution شما چند پروژه‌ای است، احتمال دارد که قسمت‌های مختلف آن از اسمبلی‌های مشابهی، اما با نگارش‌های مختلفی استفاده می‌کنند. اگر این اسمبلی‌ها از طریق نیوگت اضافه شده‌اند، دستور ذیل را صادر کنید:

```
PM> Update-Package
```

اگر خیر، فایل‌های csproj را باید تک تک بررسی کنید و شماره نگارش‌های اسمبلی‌های مشابه را تطابق دهید.
- مطلب « به روز رسانی قسمت assemblyBinding فایل‌های config توسط NuGet » را هم مدنظر داشته باشید.

نویسنده: مجتبی آزاد
تاریخ: ۱۳۹۴/۰۳/۰۹ ۱۵:۵۹

لزوما باید برای تمام نگاشت‌ها کلاس پروفایل ساخت؟
آیا امکان این وجود دارد با روشی که در بالا گفته شد، این کار به صورت خودکار انجام شود؟
اگر خیر، علت اینکه DynamicMap بدون نیاز به پروفایل به درستی عمل می‌کند چیست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۰۹ ۱۶:۱۶

وجود تنظیمات صریح در ابتدای برنامه، کار برپایی مقدمات Fast Reflection را ساده‌تر می‌کند و در نتیجه روی سرعت و کارآیی برنامه تاثیر مثبتی خواهد داشت. به این ترتیب این تنظیمات یکبار ایجاد شده و کش می‌شوند.

نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۵/۰۳ ۸:۱

با ورژن قبلی آن مشکلی وجود نداشت ولی در ورژن 4 آن اینترفیس IPlatformSpecificMapperRegistry موجود نیست . به چه شکل باید عمل کنیم برای فراخوانی متد Resolve ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۵/۰۳ ۹:۵۸

- هر زمان که نگارش 4 آن نهایی شد ، این مسایل باید بررسی شوند.
- همچنین این مسایل را هم باید با نویسنده‌ی اصلی آن مطرح کنید؛ نه اینجا.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۵/۱۹ ۱۹:۲

جهت تکمیل بحث

در نگارش 4 صرفا این دو سطر را حذف کنید:

```
var platformSpecificRegistry = PlatformAdapter.Resolve<IPlatformSpecificMapperRegistry>();
platformSpecificRegistry.Initialize();
```

این موارد به صورت توکار توسط خود AutoMapper لحاظ شده‌است و نیازی به آن‌ها نیست.
پلتفرم‌های مختلف در نگارش 4 به صورت یک اسمبلی مجزا به ازای هر پلتفرم ارائه شده‌اند و اینبار مانند قبل یکی نیستند.