عنوان: **پیاده سازی InstanceProvider برای سرویس های WCF**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۱/۱۰ ۲۰:۵
آدرس: www.dotnettips.info
گروه‌ها: WCF, Dependency Injection, InstanceProvider

اگر قصد داشته باشیم که تزریق وابستگی (Dependency Injection) را برای سرویس‌های WCF پیاده سازی کنیم نیاز به یک Instance Provider سفارشی داریم. در ابتدا باید سرویس‌های مورد نظر را در یک Ioc Container رجیستر نماییم سپس با استفاده از InstanceProvider عملیات وهله سازی از سرویس‌ها همراه با تزریق وابستگی انجام خواهد گرفت. فرض کنید سرویسی به صورت زیر داریم:

```
[ServiceBehavior( IncludeExceptionDetailInFaults = true)]
    public class BookService : IBookService
    {
        public BookService(IBookRepository bookRepository)
        {
            Repository = bookRepository;
        }

        public IBookRepository Repository
        {
            get;
            private set;
        }

        public IList<Entity.Book> GetAll()
        {
            return Repository.GetAll();
        }

    }
```

همانطور که می‌بینید برای عملیات وهله سازی از این سرویس نیاز به تزریق کلاس BookRepository است که این کلاس باید اینترفیس IBookRepository را پیاده سازی کرده باشد. برای این که Instance Provider ما بتواند عملیات تزریق وابستگی را به درستی انجام دهد، ابتدا باید BookRepository و BookService را به یک IocContainer (در این جا از الگوی ServiceLocator و UnityContainer استفاده کردم) رجیستر نماییم . به صورت زیر:

```
var container = new UnityContainer();

      container.RegisterType<IBookRepository, BookRepository>();
      container.RegisterType<BookService, BookService>();

ServiceLocator.SetLocatorProvider(new ServiceLocatorProvider(() => { return container; }));
```

حال باید InstanceProvider را به صورت زیر ایجاد نمایم:

```
public class UnityInstanceProvinder : IInstanceProvider
    {
        private Type serviceType;

        public UnityInstanceProvinder( Type serviceType )
        {
            this.serviceType = serviceType;
        }

        public object GetInstance( InstanceContext instanceContext, Message message )
        {
            return ServiceLocator.Current.GetInstance( serviceType );
        }

        public object GetInstance( InstanceContext instanceContext )
        {
            return GetInstance( instanceContext, null );
        }
```

```
        public void ReleaseInstance( InstanceContext instanceContext, object instance )
        {
            if ( instance is IDisposable )
            {
                ( ( IDisposable )instance ).Dispose();
            }
        }
    }
```

با پیاده سازی متدهای اینترفیس IInstanceProvider می‌توان عملیات وهله سازی سرویس‌های WCF را تغییر داد. متد GetInstance همین کار را برای ما انجام خواهد داد. در این متد ما با توجه به نوع ServiceType سرویس مورد نظر را از ServiceLocator تامین خواهیم کرد. چون وابستگی‌های سرویس هم در IOC Cotnainer موجود است در نتیجه سرویس به درستی وهله سازی خواهد شد. از آن جا که در WCF عملیات وهله سازی از سرویس‌ها به طور مستقیم به نوع سرویس بستگی دارد، هیچ نیازی به نوع Contract مربوطه نیست. در نتیجه Service Type به صورت مستقیم در اختیار این کلاس قرار خواهد گرفت. مرحله آخر معرفی IInstanceProvider به عنوان یک Service Behavior است. برای این کار کدهای زیر را در کلاسی به نام UnityInstanceProviderContext کپی نمایید:

```
public class UnityInstanceProviderContext : IServiceBehavior
    {
        public void AddBindingParameters( ServiceDescription serviceDescription , ServiceHostBase
serviceHostBase , Collection<ServiceEndpoint> endpoints , BindingParameterCollection bindingParameters
)
        {
        }

        public void ApplyDispatchBehavior( ServiceDescription serviceDescription , ServiceHostBase
serviceHostBase )
        {
            serviceHostBase.ChannelDispatchers.ToList().ForEach( channelDispatcherBase =>
                {
                    var channelDispatcher = ( channelDispatcherBase as ChannelDispatcher );
                    if ( channelDispatcher != null )
                    {
                        channelDispatcher.Endpoints.ToList().ForEach( endpoint =>
                            {
                                endpoint.DispatchRuntime.InstanceProvider = new UnityInstanceProvinder(
serviceDescription.ServiceType );
                            } );
                    }
                } );
        }

        public void Validate( ServiceDescription serviceDescription , ServiceHostBase serviceHostBase )
        {
        }
    }
```

در متد ApplyDispatchBehavior همان طور دیده می‌شود به ازای تمام EndPointهای هر ChannelDispatcher یک نمونه از کلاس UnityInstanceProvider به همراه پارامتر سازنده آن که نوع سرویس مورد نظر است به خاصیت InstanceProvider در DispatchRuntime معرفی می‌گردد.

در هنگام هاست سرویس مورد نظر هم باید تمام این عملیات به عنوان یک Behavior در اختیار Service Host قرار گیرد.همانند نمونه کد ذیل:

```
using (ServiceHost host = new ServiceHost(typeof(BookService)))
{
                    host.Description.Behaviors.Add( new UnityInstanceProviderContext() );
 }
```

نظرات خوانندگان

نویسنده: وحید م
تاریخ: ۱۳۹۲/۱۱/۱۱ ۸:۴۸

با سلام؛ اگر یک برنامه چند لایه داشته باشیم (UI- DomainLayer-DataAccess- Service) و مثلا یک پروژه‌ای هم از نوع webapi یا wcf در آن معماری قرار داشت، می‌خواهم در web api یا wcf هم برای اعمال dependency Injection با آن mapping ها که در لایه ui قرار دادم هم استفاده کنم. با تشکر.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۱ ۹:۳۶

در مطلب فوق محل قرارگیری container.RegisterType در نقطه آغاز برنامه است؛ جایی که نگاشت‌های مورد نیاز در سایر لایه‌ها هم انجام می‌شود. بنابراین فرقی نمی‌کند.

نویسنده: وحید م
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۰:۴۳

ممنون ولی سوال بنده کلی بود؟ وقتی یک معماری دارم بگونه ای گفته شد یا مثل cms IRIS آقای سعیدی فر و خواستم به پروژه پروژه دیگری اضافه کنم از نوع webapi یا wcf که به نوعی از لایه service هم برای اتصال به بانک استفاده میکنه DI را باید چگونه برای آن اعمال کرد ؟ آیا می‌بایست تنظیمات و mapping های داخل global مربوط به structuremap درون ui را در داخل پروژه webapi یا wcf هم قرار داد یا خیر؟ اگر webapi را جدا هاست کنیم چه تضمینی وجود دارد دیگر به پروژه webapi دسترسی نداشته باشد

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۰:۵۰

صرف نظر از اینکه برنامه شما از چند DLL نهایتا تشکیل میشه، تمام این‌ها داخل یک Application Domain اجرا می‌شن. یعنی عملا یک برنامه‌ی واحد شما دارید که از اتصال قسمت‌های مختلف با هم کار می‌کنه. IoC Container هم تنظیماتش اول کار برنامه کش می‌شه. یعنی یکبار که تنظیم شد، در سراسر آن برنامه قابل دسترسی هست. بنابراین نیازی نیست همه جا تکرار بشه. یکبار آغاز کار برنامه اون رو تنظیم کنید کافی هست.