Castle.Windsor Fluent Registration API

리얼웹	배성혁	2011-11-08 오후 1:59

Castle.Windsor 는 IoC/DI 라이브러리로 유명한 제품 중에 하나이고, 리얼웹에서는 가장 핵심적인 요소로 자리 잡고 있습니다. 앞으로는 AOP (Aspect Oriented Programming)로 확장할 수 도 있습니다.

Castle.Windsor 는 IoC/DI 뿐 아니라 DynamicProxy 를 이용하여, Interceptor 로 감싼 Proxy를 제공하여, Container에서 제공하는 Component 가 실제 원하는 인스턴스가 아닌 Interceptor 로 감싸진 Proxy를 전달 받아서 사용하는 방법도 있습니다. 이 방법은 WPF, Silverlight 등에서 View Model 용 Class가 INotifyPropertyChanged, IEditableObject 에 대한 구현을 수행하지 않더라도, 자동으로 구현해주는 기능도 제공합니다.

Castle.Windsor 에서 Component 정의하고, 등록하는 방법에는 여러 가지 방식이 있는데, 첫째 xml 파일에 등록할 component 를 정의하는 것이고, 둘째, 코드상으로 Component 를 등록하는 것입니다.

각 방식의 특징을 살펴 본다면,

1. Xml 파일 사용 (Using XML configuration)
   a. 환경설정처럼 사용되므로, 관리자가 편집기를 통해 수정하여 바로 적용. 컴파일이 필요 없다.
   b. 여러 Use Case에 해당하는 Xml 파일들을 준비하고, 필요한 것만 선택적으로 적용할 수 있다.
      특히 installer 를 이용하면 좋음 (참고:Registering Installers)
   c. 아주 많은 컴포넌트를 정의하려면, 작성 시 많은 노력이 필요하다.
   d. 제대로 작성되었는지는 Runtime 시에만 확인이 가능하다. (철자 하나 틀리면 허무함)

2. 코드로 Component 등록하기 (Fluent Registration API)
   a. 대량의 Component를 한방에 등록할 수 있다. (CoponentId 는 Component Type의 FullName 이다)
      특정 Assembly 의 모든 수형을 등록!!! (xml 방식으로는 절대 불가능)
   b. 코드에서 정의하므로, 작성 중, 컴파일 시에 잘못된 설정을 걸러낼 수 있다.
   c. 동적으로 Component를 등록할 수 있다. (특히 이미 등록 여부에 따라 등록을 제한하는 경우 등)
   d. Conditional Compiler 지시자를 통해, 선택적으로 코드를 실행할 수 있다.

XML Configuration 을 이용한 방식은 이미 많이 사용하고 있고, 익숙하리라 생각되어, 이번에는 Fluent Registration API 에 대해 설명하면서, 대응되는 XML Configuration을 도시하여, 이해를 높히도록 하겠습니다.

1. 모든 컴포넌트 등록하기

```
// ICompressor를 정의한 Assembly에서 ICompressor를 구현한 모든 클래스를 등록합니다.
_container.Register(AllTypes
                    .FromAssemblyContaining<ICompressor>()
                    .BasedOn<ICompressor>());

// WithService.DefaultInterface() 는 수형이 ICompressor를 구현하지만, 다른 인터페이스도 구현한다면,
// 그것도 등록해 준다. (7. MultiService 등록 참고 )
_container.Register(AllTypes
                    .FromAssemblyContaining<ICompressor>()
                    .BasedOn<ICompressor>()
                    .WithService.DefaultInterface());
```

2. 특정 컴포넌트 등록

a. Fluent Registration

```
// 특정 Service를 구현한 Class를 지정하여 등록합니다.
_container.Register(
        Component
                .For<ICompressor>()
                .ImplementedBy<SharpBZip2Compressor>()
                .Named("SharpBZip2")
                .LifeStyle.Singleton,
```

```
        Component
                .For(typeof(ISerializer<>))
                .ImplementedBy(typeof(CompressSerializer<>))
                .Named("CompressSerializer.Generic")
                .LifeStyle.PerThread);
_container.ResolveAll<ICompressor>().Count().Should().Be(1);
_container.Resolve<ISerializer<UserInfo>>("CompressSerializer.Generic").Should().Not.Be.Null();
```

b. Xml Format

```
<component id="SharpBZip2Compressor"
           service="RCL.Core.ICompressor, RCL.Core"
           type="RCL.Core.SharpBZip2Compressor, RCL.Core"
           lifestyle="Singleton"/>
<component id="CompressSerializer.Generic"
           service="RCL.Core.ISerializer`1, RCL.Core"
           type="RCL.Core.RwBsonSerializer`1, RCL.Core">
</component>
```

3. 특정 인스턴스를 Container에 컴포넌트로 등록

a. 이미 생성된 인스턴스를 컴포넌트로 등록합니다. 특정 Factory에 의해 성성되었거나, 외부에서 제공된 인스턴스를 계속 활용하기 위한 방편입니다.

```
var compressor = new SharpBZip2Compressor();
_container.Register(Component
                    .For<ICompressor>()
                    .Instance(compressor));
```

4. 인스턴스 Factory를 이용하여 컴포넌트 등록

a. 컴포넌트를 지정하는 것이 아니라, 컴포넌트에 해당하는 수형의 인스턴스를 생성하는 델리게이트를 등록하여, 컨테이너가 인스턴스를 생성하게끔 합니다.

```
_container
        .AddFacility<FactorySupportFacility>()
        .Register(Component
                    .For<ICompressor>()
                    .UsingFactoryMethod(() => new SharpBZip2Compressor()));
_container.Resolve<ICompressor>().Should().Be.InstanceOf<SharpBZip2Compressor>();
```

5. 컴포넌트 생성 시 속성 값 설정

a. 컴포넌트 생성 시 속성에 값을 설정하는 방식에 해당합니다.
단 이 예에서도 보였듯이, 코드상으로 정의하게 되면, 속성 값을 단정된 값이 아닌 ValueFactory를 이용하여, Component 생성 시마다 다른 설정 값을 줄 수도 있습니다. (XML에서는 불가능합니다)

```
Func<DateTime> @getDate = () => DateTime.Today;
_container
        .Register(Component
                    .For<ITimeBlock>()
                    .ImplementedBy<TimeBlock>()
                    .OnCreate((kernel, instance) =>
                            {
                                    instance.Start = @getDate();
                                    instance.Duration = TimeSpan.FromDays(1);
                            })
                    .LifeStyle.Transient);
var block = _container.Resolve<ITimeBlock>();
var today = @getDate();
block.Start.Should().Be(today);
block.Duration.Should().Be(TimeSpan.FromDays(1));
block.Should().Be(new TimeBlock(today, TimeSpan.FromDays(1)));
```

b. XML로 속성이나 생성자의 인자 값을 설정하려면, parameters element 내에 해야 합니다.

```
<component id="IniAdoQueryStringProvider.Northwind"
           service="RCL.Data.IQueryStringProvider, RCL.Data"
           type="RCL.Data.IniQueryStringProvider.IniAdoQueryStringProvider, RCL.Data"
                  lifestyle="singleton">
        <parameters>
                <queryFilePath>.\QueryFiles\Northwind.Ado.MsSql.ini</queryFilePath>
        </parameters>
</component>
```

6. 속성 및 Dependency Injection 설정 하기

a. 컴포넌트의 속성 중에 다른 컴포넌트를 Injection 할 때 사용합니다.

```
//! new CompressorSerializer<T>( new RwJsonByteSerializer<T>(), new SharpBZip2Compressor() );
_container.Register(
        Component.For<ICompressor>()
            .ImplementedBy<SharpBZip2Compressor>(),
        Component
            .For(typeof(ISerializer<>))
            .ImplementedBy(typeof(RwJsonByteSerializer<>)),
// 위의 두 컴포넌트를 속성으로 Dependency Injection 을 수행합니다.

        Component
            .For(typeof(ISerializer<>))
            .ImplementedBy(typeof(CompressSerializer<>))
            .Named("CompressSerializer")
            .ServiceOverrides(
                  ServiceOverride
                      .ForKey("serializer")
                      .Eq("RCL.Core.RwJsonByteSerializer`1"),
                  ServiceOverride
                      .ForKey("compressor")
                      .Eq("RCL.Core.SharpBZip2Compressor")));

var component = _container.Resolve<ISerializer<UserInfo>>("CompressSerializer");
component.Should().Not.Be.Null();
component.Should().Be.InstanceOf<CompressSerializer<UserInfo>>();
var compressSerializer = (CompressSerializer<UserInfo>)component;
compressSerializer.Serializer.Should().Not.Be.Null();
compressSerializer.Serializer.Should().Be.InstanceOf<RwJsonByteSerializer<UserInfo>>();
compressSerializer.Compressor.Should().Not.Be.Null();
compressSerializer.Compressor.Should().Be.InstanceOf<SharpBZip2Compressor>();
```

b. XML 로 정의

```
<component id="Serializer.Northwind"
           service="RCL.Core.ISerializer`1, RCL.Core"
           type="RCL.Core.CompressSerializer`1, RCL.Core">
        <parameters>
               <serializer>${JsonSerializer}</serializer>
               <compressor>${Compressor.SharpBZip2}</compressor>
        </parameters>
</component>

<component id="JsonSerializer"
          service="RCL.Core.ISerializer`1, RCL.Core"
          type="RCL.Core.RwJsonByteSerializer`1, RCL.Core">
</component>
<component id="Compressor.SharpBZip2"
          service="RCL.Core.ICompressor, RCL.Core"
          type="RCL.Core.SharpBZip2Compressor, RCL.Core" />
```

c. IAdoRepository 등록하기 (dbName, QueryStringProvider 정의)

```
container.Register(
    Component
        .For<IQueryStringProvider>()
        .ImplementedBy<IniAdoQueryStringProvider>()
        .Named("IniQueryStringProvider.Northwind")
        .Parameters(Parameter
                        .ForKey("queryFilePath")
                        .Eq(@"QueryFiles\Northwind.ado.mssql.ini")),
    Component
        .For<IAdoRepository>()
        .ImplementedBy<AdoRepositoryImpl>()
        .Named("AdoRepository.Northwind")
        .Parameters(Parameter
                        .ForKey("dbName")
                        .Eq("Northwind"))
        .ServiceOverrides(ServiceOverride
                              .ForKey("queryStringProvider")
                              .Eq("IniQueryStringProvider.Northwind"))
);
```

d. IAdoRepository 를 XML 로 등록하기

```
<component id="NorthwindAdoRepository"
           service="RCL.Data.Ado.IAdoRepository, RCL.Data"
           type="RCL.Data.Ado.AdoRepositoryImpl, RCL.Data"
           lifestyle="singleton">
```

```
        <parameters>
            <!-- dbName을 설정하지 않으면 DAAB의 기본 DB 사용 -->
            <!--<dbName>Northwind</dbName>-->
            <QueryStringProvider>${IniAdoQueryStringProvider.Northwind}</QueryStringProvider>
        </parameters>
</component>

<component id="IniAdoQueryStringProvider.Northwind"
           service="RCL.Data.IQueryStringProvider, RCL.Data"
           type="RCL.Data.IniQueryStringProvider.IniAdoQueryStringProvider, RCL.Data"
                 lifestyle="singleton">
        <parameters>
                <queryFilePath>.\QueryFiles\Northwind.Ado.MsSql.ini</queryFilePath>
        </parameters>
</component>
```

7. Multi Service에 대응하는 컴포넌트 등록

a. 복수의 서비스 인터페이스에 대해 하나의 Concrete Class로 대응되도록 수형을 등록합니다.

```
_container.Register(
        Component
            .For<ITimePeriod, ITimeRange>()
            .ImplementedBy<TimeRange>());
_container.Resolve<ITimePeriod>().Should().Be.InstanceOf<TimeRange>();
_container.Resolve<ITimeRange>().Should().Be.InstanceOf<TimeRange>();
```

8. Service Forwarding

a. 특정 서비스의 컴포넌트를 다른 서비스에서도 사용가능하도록 포워드합니다.

```
_container.Register(
        Component.For<ITimeRange>()
                .Forward<ITimePeriod, IComparable>()
                .ImplementedBy<TimeRange>());
_container.Resolve<ITimePeriod>().Should().Be.InstanceOf<TimeRange>();
_container.Resolve<IComparable>().Should().Be.InstanceOf<TimeRange>();
_container.Resolve<ITimeRange>().Should().Be.InstanceOf<TimeRange>();
```

9. Unless - 특정 조건이 만족하지 않을 때에만 등록

a. 이미 등록된 컴포넌트가 아닐 때에만 등록 (예외가 발생하지 않습니다.)

```
// 등록
_container.Register(
        Component.For(typeof(ISerializer<>))
                .ImplementedBy(typeof(RwJsonByteSerializer<>))
                .Unless(Component.ServiceAlreadyRegistered));
// 등록 안 함
_container.Register(
        Component.For(typeof(ISerializer<>))
                .ImplementedBy(typeof(RwCompressJsonSerializer<>))
                .Unless(Component.ServiceAlreadyRegistered));
```

b. Lambda Expression 또는 Predicate<bool> 를 이용하여 표현 가능

```
// 특정 수형(ITimePeriod)을 구현한 클래스 중에
// TimePeriodBase로 Casting 할 수 없는 수형만 등록합니다.
_container.Register(
        AllTypes
                .FromAssemblyContaining<ITimePeriod>()
                .BasedOn(typeof(ITimePeriod))
                .Unless(t => typeof(TimePeriodBase).IsAssignableFrom(t)));
var components = _container.ResolveAll<ITimePeriod>();
components.Length.Should().Be.GreaterThan(0);
components.RunEach(c => Console.WriteLine(c.GetType().FullName));
```

10. If - 특정 조건이 만족할 때만 등록

a. Unless 와 반대의 경우

```
// 특정 수형(ITimePeriod)을 구현한 클래스 중에 TimePeriodBase로 Casting 할 수 있는 수형만 등록합니다.
```

```
_container.Register(
        AllTypes
                .FromAssemblyContaining<ITimePeriod>()
                .BasedOn(typeof(ITimePeriod))
                .If(t => typeof(TimePeriodBase).IsAssignableFrom(t)));

var components = _container.ResolveAll<ITimePeriod>();
components.Length.Should().Be.GreaterThan(0);
components.RunEach(c => Console.WriteLine(c.GetType().FullName));

// 특정 수형(ITimePeriod)을 구현한 클래스 중에 수형 중에 "Block" 이란 명칭이 들어간 수형만 등록합니다.
// 여기서는 TimeBlock 하나만!!!
_container.Register(
        AllTypes
                .FromAssemblyContaining<ITimePeriod>()
                .BasedOn(typeof(ITimePeriod))
                .If(t => t.FullName.Contains("Block")));
var components = _container.ResolveAll<ITimePeriod>();
components.Length.Should().Be.GreaterThan(0);
components.RunEach(c => Console.WriteLine(c.GetType().FullName));
```

11. Interceptor 등록 (Registering Interceptors and ProxyOptions)

a. Castle.Windsor의 장점 중 하나인 컴포넌트에 Interceptor를 추가하여, Proxy 로 제공하는 기능을 구현한 것입니다.

```
_container.Register(
        Component
                .For(typeof(ISerializer<>))
                .Interceptors(InterceptorReference.ForType<NotifyPropertyChangedInterceptor>(),
                              InterceptorReference.ForType<EditableObjectInterceptor>())
                .Last,
        Component
                .For<ISerializer>()
                .Interceptors(InterceptorReference.ForType<EditableObjectInterceptor>())
                .Last,
Component.For<NotifyPropertyChangedInterceptor>(),
        Component.For<EditableObjectInterceptor>());
var serializer = _container.Resolve(typeof(ISerializer<UserInfo>));
serializer.Should().Not.Be.Null();
// interceptor 媛앹 ?뚯썝吏?nbsp;Proxy?댁? ?대뙏?⑸땲??
serializer.IsDynamicProxy().Should().Be.True();
```

이와 같이 구현한다면, serializer 는 Proxy 객체이고, INotifyPropertyChanged, IEditableObject 인터페이를 가지게 됩니다.

12. Interceptor Proxy MixIn

a. Interceptor 를 직접 등록하는 것이 아니라 AOP 방식으로 엮는 경우가 있습니다. Class를 Interceptor 와 엮는 역할을 해 줍니다.

```
_container.Register(Component.For<ICompressor>()
                             .ImplementedBy<SharpBZip2Compressor>()
                             .Proxy.MixIns(new EditableObjectInterceptor()));
var component = _container.Resolve<ICompressor>();
component.Should().Not.Be.Null();
component.IsDynamicProxy().Should().Be.True();
Console.WriteLine(component.GetType().FullName);
```

13. Fluent Registration API Extensions

a. 공부 중

14. IWindsorInstaller 를 이용한 등록
15. RCL.DataServices.WebHost 에서 사용하는 예

1. RCL.DataServices 예제

a. AdoRepositoryWindsorInstaller

```
public class AdoRepositoryWindsorInstaller : IWindsorInstaller
{
     public void Install(IWindsorContainer container, IConfigurationStore store)
```

```
        {
            container.Register(

               Component
                   .For<IAdoRepository>()
                   .ImplementedBy<SqlRepositoryImpl>()
                   .Named("AdoRepository.Northwind")
                   .Parameters(Parameter.ForKey("dbName").Eq("Northwind"))
                   .ServiceOverrides(ServiceOverride
                                   .ForKey("queryStringProvider")
                                         .Eq("QueryStringProvider.Northwind")),

               Component
                   .For<IAdoRepository>()
                   .ImplementedBy<SqlRepositoryImpl>()
                   .Named("AdoRepository.Pubs")
                   .Parameters(Parameter.ForKey("dbName").Eq("Pubs"))
                   .ServiceOverrides(ServiceOverride
                                         .ForKey("queryStringProvider")
                                         .Eq("QueryStringProvider.Pubs")),

               Component
                      .For<IQueryStringProvider>()
                      .ImplementedBy<IniAdoQueryStringProvider>()
                      .Named("QueryStringProvider.Northwind")
                      .Parameters(Parameter
                                      .ForKey("queryFilePath")
                                      .Eq(@"QueryFiles\Northwind.ado.mssql.ini")),

               Component
                   .For<IQueryStringProvider>()
                      .ImplementedBy<IniAdoQueryStringProvider>()
                      .Named("QueryStringProvider.Pubs")
                      .Parameters(Parameter
                                      .ForKey("queryFilePath")
                                      .Eq(@"QueryFiles\Pubs.ado.mssql.ini"))
             );
        }
}
```

b. DataServiceWindsorInstaller

```
public class DataServiceWindsorInstaller : IWindsorInstaller
{
    public void Install(IWindsorContainer container, IConfigurationStore store)
    {
       container.Register(
           Component
               .For<IDataService>()
               .ImplementedBy<AsyncDataServiceImpl>()
               .Named("DataService.Northwind")
               .ServiceOverrides(ServiceOverride
                                   .ForKey("AdoRepository")
                                   .Eq("AdoRepository.Northwind"),
                                ServiceOverride
                                   .ForKey("NameMapper")
                                   .Eq("RCL.Data.Ado.TrimNameMapper")),
           Component
               .For<IDataService>()
               .ImplementedBy<AsyncDataServiceImpl>()
               .Named("DataService.Pubs")
               .ServiceOverrides(ServiceOverride
                                    .ForKey("AdoRepository")
                                    .Eq("AdoRepository.Pubs"),
                                ServiceOverride
                                    .ForKey("NameMapper")
                                    .Eq("RCL.Data.Ado.TrimNameMapper")),
           Component
               .For<IDataServiceAdapter>()
               .ImplementedBy<DataServiceAdapter>()
               .Named("DataServiceAdapter.Northwind")
               .ServiceOverrides(ServiceOverride
                                   .ForKey("DataService")
                                   .Eq("DataService.Northwind"),
                                ServiceOverride
                                   .ForKey("RequestSerializer")
                                   .Eq("MessageSerializer`
           1[[RCL.DataServices.Messages.RequestMessage, RCL.DataServices.Messages]]"),
                                ServiceOverride
                                       .ForKey("ResponseSerializer")
                                       .Eq("MessageSerializer`
           1[[RCL.DataServices.Messages.ResponseMessage, RCL.DataServices.Messages]]")),

           Component
               .For<IDataServiceAdapter>()
               .ImplementedBy<DataServiceAdapter>()
```

```
                .Named("DataServiceAdapter.Pubs")
                .ServiceOverrides(ServiceOverride
                                      .ForKey("DataService")
                                      .Eq("DataService.Pubs"),
                                  ServiceOverride
                                      .ForKey("RequestSerializer")
                                      .Eq("MessageSerializer`
            1[[RCL.DataServices.Messages.RequestMessage, RCL.DataServices.Messages]]"),
                                  ServiceOverride
                                      .ForKey("ResponseSerializer")
                                      .Eq("MessageSerializer`
            1[[RCL.DataServices.Messages.ResponseMessage, RCL.DataServices.Messages]]"))
            );
    }
}
```

c. MessageSerializerWindsorInstaller

```
public class MessageSerializerWindsorInstaller : IWindsorInstaller
{
    public void Install(IWindsorContainer container, IConfigurationStore store)
    {
        container.Register(
            Component
                    .For(typeof(ISerializer<>))
                    .ImplementedBy(typeof(CompressSerializer<>))
                    .Named("MessageSerializer.Northwind")
                    .ServiceOverrides(ServiceOverride
                                          .ForKey("serializer")
                                          .Eq("RCL.Core.RwJsonByteSerializer`1"),
                                      ServiceOverride
                                          .ForKey("compressor")
                                          .Eq("RCL.Core.SharpBZip2Compressor")),

            Component
                    .For(typeof(ISerializer<>))
                    .ImplementedBy(typeof(EncryptSerializer<>))
                    .Named("MessageSerializer.Pub")
                    .ServiceOverrides(ServiceOverride
                                          .ForKey("serializer")
                                          .Eq("CompressSerializer.Pub"),
                                      ServiceOverride
                                          .ForKey("encryptor")
                                          .Eq("RCL.Core.AriaSymmetricEncryptor")),

            Component
                    .For(typeof(ISerializer<>))
                    .ImplementedBy(typeof(CompressSerializer<>))
                    .Named("CompressSerializer.Pub")
                    .ServiceOverrides(ServiceOverride
                                          .ForKey("serializer")
                                          .Eq("RCL.Core.RwBsonSerializer`1"),
                                      ServiceOverride
                                          .ForKey("compressor")
                                          .Eq("RCL.Core.SharpBZip2Compressor"))
            );
    }
}
```

d. Container Install

```
if(IoC.IsNotInitialized)
{
    var container = new WindsorContainer();

    container.Install(FromAssembly.This(),
                      FromAssembly.Containing<ICompressor>(),
                      FromAssembly.Containing<INameMapper>());

    IoC.Initialize(container);
}
```

e. Container Install with FutureIoCHttpApplication

```
public class DataServiceHttpApplication : FutureIoCHttpApplication
{
    protected override IWindsorContainer SetUpContainer()
    {
        var container = new WindsorContainer();

        container.Install(FromAssembly.This(),
                          FromAssembly.Containing<ICompressor>(),
                          FromAssembly.Containing<INameMapper>());
        return container;
```

```
    }
}
```