SOLID Design Principles & Implementations

By:
Noha Ahmed Thabet

Demo custom ioc container

IOC Containers

All the containers must provide easy support for the following DI lifecycle.

- **Register:**

The container must know which dependency to instantiate when it encounters a particular type. This process is called registration.

- **Resolve:**

When using the IoC container, we don't need to create objects manually. The container does it for us. This is called resolution.
The container must include some methods to resolve the specified type; the container creates an object of the specified type, injects the required dependencies if any and returns the object.

- **Dispose:**

The container must manage the lifetime of the dependent objects. Most IoC containers include different **LifetimeManagers** to manage an object's lifecycle and dispose it.

IOC Containers

- There are many open source or commercial containers available for .NET. Some are listed below.

- StructureMap
- Castle Windsor
- Ninject
- Autofac
- DryIoc
- Simple Injector
- Light Inject
- Unity

Introduction to Windsor

- **What is it?**
  - Open source IoC container
  - Part of the large Castle project
  - One of the first IoC containers for .NET

Register Types Via Code

```
WindsorContainer container = new WindsorContainer();

container.Register(Component.For<Shoper>().ImplementedBy<Shoper>());
container.Register(Component.For<ICreditCard>().ImplementedBy<VisaCard>());

Shoper shoper = container.Resolve<Shoper>();
```

Register Using Configuration

```
WindsorContainer container = new WindsorContainer("castle.config");
Shoper shoper = container.Resolve<Shoper>();
```

Castle.config

```
<configuration>
  <components>
    <component
    service="IocContainer.Shoper, IocContainer"
    type="IocContainer.Shoper, IocContainer"/>
    <component
    service="IocContainer.ICreditCard, IocContainer"
    type="IocContainer.MasterCard, IocContainer" />
  </components>
</configuration>
```

Features

```
 //1 using names for registrations
container.Register(Component.For<ICreditCard>().ImplementedBy<Visa
Card>().Named("visa"));
var card = container.Resolve<ICreditCard>("visa");

//2- register instance (Singleton)
VisaCard card = new VisaCard();
container.Register(Component.For<ICreditCard>().Instance(card));

//3- Setter Injection is done automatically
```

Lifecycle Management

```
//1 Trasient
container.Register(Component.For<Shoper>().LifeStyle.Transient);

//2- Singleton (Default)
container.Register(Component.For<Shoper>().LifeStyle.Singleton);

//3- Per thread
container.Register(Component.For<Shoper>().LifeStyle.PerThread);

//4- Per web request (good for web apps)
container.Register(Component.For<Shoper>().LifeStyle.PerWebRequest);
```

Demo Castel Wendsor ioc container

Introduction to Unity

- **What is it?**
  - IoC container from Microsoft
  - Part of Enterprise Library
  - Also available as NuGet package

Register Types Via Code

```
UnityContainer container = new UnityContainer();
container.RegisterType<Shoper, Shoper>();
container.RegisterType<ICreditCard, DebitCard>();

Shoper shoper = container.Resolve<Shoper>();
```

Register Using Configuration

```
<configuration>
  <configSections>
    <section name="unity"
type="Microsoft.Practices.Unity.Configuration.UnityConfiguration
Section, Microsoft.Practices.Unity.Configuration"/>
  </configSections>
  <unity
xmlns="http://schemas.microsoft.com/practices/2010/unity">
    <container>
      <register type="IocContainer.Shoper, IocContainer"
                mapTo="IocContainer.Shoper, IocContainer"/>
      <register type="IocContainer.ICreditCard, IocContainer"
                mapTo="IocContainer.VisaCard, IocContainer"/>
    </container>
  </unity>
</configuration>
```

Features

```
//1- using names for registerations
container.RegisterType<ICreditCard, MasterCard>("master");
container.RegisterType<ICreditCard, MasterCard>(“visa");
var card = container.Resolve<ICreditCard>("master");


//2- register instance (Singleton)
MasterCard masterCard = new MasterCard();
container.RegisterInstance<ICreditCard>(masterCard);


//3- Overide Registered Items
container.RegisterType<ICreditCard, DebitCard>();
Shoper shoper = container.Resolve<Shoper>(
            new ParameterOverride("card", new VisaCard()));
```

Lifecycle Management

```
 //1- Transient (Default)
container.RegisterType<ICreditCard, DebitCard>(new
    TransientLifetimeManager());

//2- Singleton
container.RegisterType<ICreditCard, DebitCard>(new
    ContainerControlledLifetimeManager());

//3- Per thread
container.RegisterType<ICreditCard, DebitCard>(new
    PerThreadLifetimeManager());
```

Demo Unity ioc container