‹ Previous Next ›

Unity Container: Property Injection

In the previous chapter, we learned about constructor injection. Here, we will learn about property injection using Unity Container.

Property injection is a type of dependency injection where dependencies are provided through properties. Visit the Dependency Injection chapter to learn more about it.

Let's understand how we can perform property injection using Unity container. Consider the following example classes.

Example: C# Copy

```
public interface ICar
{
    int Run();
}

public class BMW : ICar
{
    private int _miles = 0;

    public int Run()
    {
        return ++_miles;
    }
}

public class Ford : ICar
{
    private int _miles = 0;

    public int Run()
    {
        return ++_miles;
    }
}

public class Audi : ICar
{
    private int _miles = 0;

    public int Run()
    {
        return ++_miles;
    }

}
public class Driver
{
    public Driver()
    {
    }

    [Dependency]
    public ICar Car { get; set; }

    public void RunCar()
```

```
    {
        Console.WriteLine("Running {0} - {1} mile ",
                          this.Car.GetType().Name, this.Car.Run());
    }
}
```

As you can see in the above sample classes, the `Driver` class is dependent on a property of type `ICar`. So, we need to set an object of a class that implements `ICar` to the `Car` property using Unity container.

Property injection in Unity container can be implemented in two ways:

1. Using the [Dependency] attribute
2. Using run-time configuration

Dependency Attribute

For the property injection, we first tell the Unity container which property to inject. So, we need to decorate the dependent properties with the `[Dependency]` attribute, as shown in the following `Driver` class.

Example: Apply [Dependency] Attribute - C# Copy

```
public class Driver
{

    public Driver()
    {
    }

    [Dependency]
    public ICar Car { get; set; }

    public void RunCar()
    {
        Console.WriteLine("Running {0} - {1} mile ", this.Car.GetType().Name, this.Car.Run());
    }
}
```

Now, we can register the `ICar` type and resolve it as shown below.

Example: Property Injection using Unity Container - C# Copy

```
var container = new UnityContainer();
container.RegisterType<ICar, BMW>();

var driver = container.Resolve<Driver>();
driver.RunCar();
```

Output:

```
Running BMW - 1 mile
```

Named Mapping

We can specify a name in the `[Dependency("name")]` attribute, which can then be used to set the property value.

```
public class Driver
{
    public Driver()
    {
    }

    [Dependency("LuxuryCar")]
    public ICar Car { get; set; }

    public void RunCar()
    {
        Console.WriteLine("Running {0} - {1} mile ", this.Car.GetType().Name, this.Car.Run());
    }
}
```

So now, we can resolve it as below.

Example: Property Injection using Unity Container - C# Copy

```
var container = new UnityContainer();
container.RegisterType<ICar, BMW>();
container.RegisterType<ICar, Audi>("LuxuryCar");

var driver = container.Resolve<Driver>();
driver.RunCar();
```

Output:

```
Running Audi - 1 mile
```

Run-time Configuration

Unity container allows us to configure a property injection with the `RegisterType()` method if a method is not marked with the `[Dependency]` attribute. You can pass an object of the InjectionProperty class in the RegisterType() method to specify a property name and a parameter value.

Note:

InjectionProperty is derived from the InjectionMember Class. The InjectionMember is an abstract class which can be used to configure injection type. There are three subclasses of InjectionMembers: InjectionConstruction to configure construction injection, InjectionProperty to configure property injection and InjectionMethod to configure method injection.

```
var container = new UnityContainer();

//run-time configuration
container.RegisterType<Driver>(new InjectionProperty("Car", new BMW()));

var driver = container.Resolve<Driver>();
driver.RunCar();
```

Output:

```
Running BMW - 1 Mile
```

As you can see in the above example, `container.RegisterType<driver>(new InjectionProperty("Car", new BMW()))` registers the `Driver` class by passing an object of InjectionProperty that specifies the property name "Car" and the `BMW` object as a value. Therefore,

Unity container will set an object of `BMW` to the `Car` property when we resolve it using `container.Resolve<Driver>()`.

‹ Previous Next ›