‹ Previous Next ›

Unity Container: Overrides

We have seen in the previous chapters that the Unity container injects the registered type by default every time we resolve the specified type. But what if we want to inject different types other than the registered type?

Unity container allows us to override a registered type by using `ResolverOverride`. The `ResolverOverride` is an abstract class that provides implementation for overriding registration. There are three important classes which inherit `ResolverOverride`:

1. ParameterOverride: Used to override constructor parameters.
2. PropertyOverride: Used to override the value of a specified property.
3. DependencyOverride: Used to override the type of dependency and its value.

Let's understand each override using 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
{
    private ICar _car = null;
```

```
    public Driver(ICar car)
    {
        _car = car;
    }

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

ParameterOverride

The ParameterOverride can be used to override registered construction parameter values.

The following example demonstrates overriding a constructor parameter.

Example: ParameterOverride - C# Copy

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

var driver1 = container.Resolve<Driver>(); // Injects registered ICar type
driver1.RunCar();

// Overrides the registered ICar type
var driver2 = container.Resolve<Driver>(new ParameterOverride("car", new Ford()));
driver2.RunCar();
```

Output:

```
Running BMW - 1 Mile
Running Ford - 1 Mile
```

In the above example, Unity container injects BMW in `driver1` which is default mapping. However, we override the default mapping and specify a different mapping for `driver2` by passing `new ParameterOverride("car", new Ford())` into the `Resolve()` method. The first parameter is the name of the constructor parameter and the second is the value of the parameter. So, `driver2` includes an object of the `Ford` class instead of the `BMW` class.

If a constructor includes multiple parameters, then we can override them by passing an array of `ResolverOverride` as shown below.

Example: Override Multiple Parameters - C# Copy

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

var driver1 = container.Resolve<Driver>();
driver1.RunCar();

var driver2 = container.Resolve<Driver>( new ResolverOverride[] {
        new ParameterOverride("car1", new Ford()),
        new ParameterOverride("car2", new BMW()),
        new ParameterOverride("car3", new Audi())
});
driver2.RunCar();
```

PropertyOverride

We learned about Property Injection in the previous chapter. Here, we will learn how to override the registered value of the specified property using PropertyOverride.

You can override a registered property injection and provide a different property value when you resolve it.

Example: PropertyOverride - C# Copy

```
var container = new UnityContainer();

//Configure the default value of the Car property
container.RegisterType<Driver>(new InjectionProperty("Car", new BMW()));

var driver1 = container.Resolve<Driver>();
driver1.RunCar();

//Override the default value of the Car property
var driver2 = container.Resolve<Driver>(
    new PropertyOverride("Car", new Audi()
);

driver2.RunCar();
```

Output:

```
BMW - 1 mile
Audi - 1 mile
```

DependencyOverride

The `DependencyOverride` class can be used to override the type of dependency and its value, irrespective of whether dependencies are provided through a constructor, a property or a method.

You can override a registered method injection and provide a different parameter value when you resolve it. Consider the following example.

Example: DependencyOverride - C# Copy

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

var driver1 = container.Resolve<Driver>();
driver1.RunCar();

//Override the dependency
var driver2 = container.Resolve<Driver>(new DependencyOverride<ICar>(new Audi())
driver2.RunCar();
```

Output:

```
Running BMW - 1 mile
Running Audi - 1 mile
```

Learn more about resolving Objects by Using Overrides.

Share Tweet Share Whatsapp

‹ Previous Next ›

TUTORIALSTEACHER.COM

TutorialsTeacher.com is optimized for learning web technologies step by step. Examples might be simplified to improve reading and basic understanding. While using this site, you agree to have read and accepted our terms of use and privacy policy.

feedback@tutorialsteacher.com

TUTORIALS

- ASP.NET Core
- ASP.NET MVC
- IoC
- Web API
- C#
- LINQ
- Entity Framework
- AngularJS 1
- Node.js
- D3.js
- JavaScript
- jQuery
- Sass
- Https

E-MAIL LIST

Subscribe to TutorialsTeacher email list and get latest updates, tips & tricks on C#, .Net, JavaScript, jQuery, AngularJS, Node.js to your inbox.

Email address GO

We respect your privacy.

HOME PRIVACY POLICY TERMS OF USE ADVERTISE WITH US

© 2020 TutorialsTeacher.com. All Rights Reserved.