**Reuse.Transient**

**Means no reuse:** transient service will be created each time when resolved or injected. Transient is the default if you omit the reuse parameter in registration and don't change Container.Rules.DefaultReuse.

The following two registrations are the same:

container.Register<IFoo, Foo>(Reuse.Transient);

container.Register<IFoo, Foo>();

### **Disposable Transient**

When you register transient service implementing IDisposable interface it becomes problematic who is responsible for service disposal. DryIoc does not support transient disposable registration by default (explained later). You need to use allowDisposableTransient: true to setup an individual registration

container.Register<X>(setup: Setup.With(allowDisposableTransient: true));

// You are in control to dispose, because you have an access to `x`

var x = container.Resolve<X>();

x.Dispose();

## Reuse.Scoped

container.Register<Car>(Reuse.Scoped);

Car car = container.Resolve<Car>()

## Reuse.Singleton

container.Register<A>(Reuse.Singleton);

var a = container.Resolve<A>();

Reference: <https://github.com/dadhi/DryIoc/blob/master/docs/DryIoc.Docs/ReuseAndScopes.md#reusetransient>