

Common Pitfalls and Challenges



Henry Been

Independent DevOps & Azure Architect

linkedin.com/in/henrybeen | henrybeen.nl

Overview



One implementing type, two service types

Hanging state

Expensive dependencies

Constructor parameters

IDisposable



Demo



Reusing the same implementing type for multiple service types



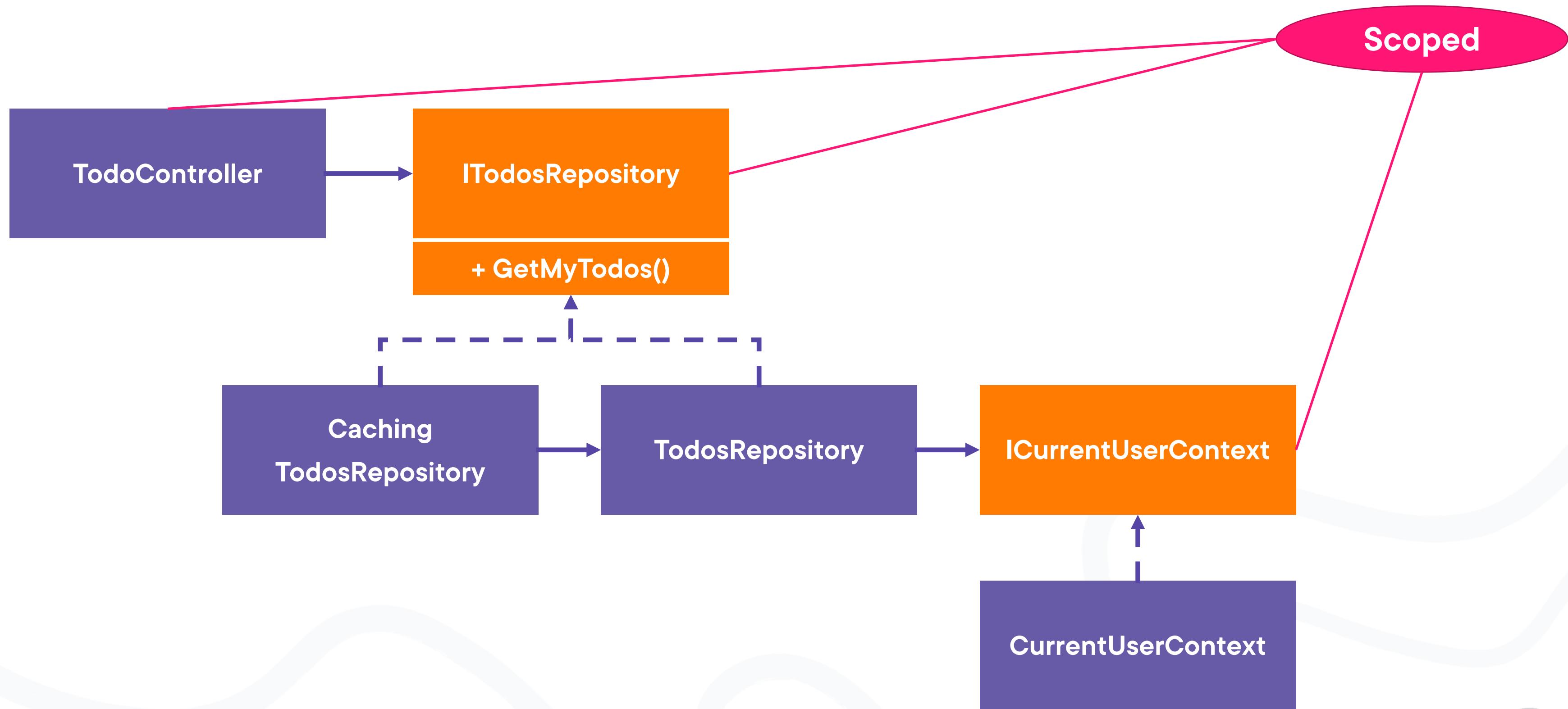
**Lifetimes are coupled to the
service type, not the
implementing type**



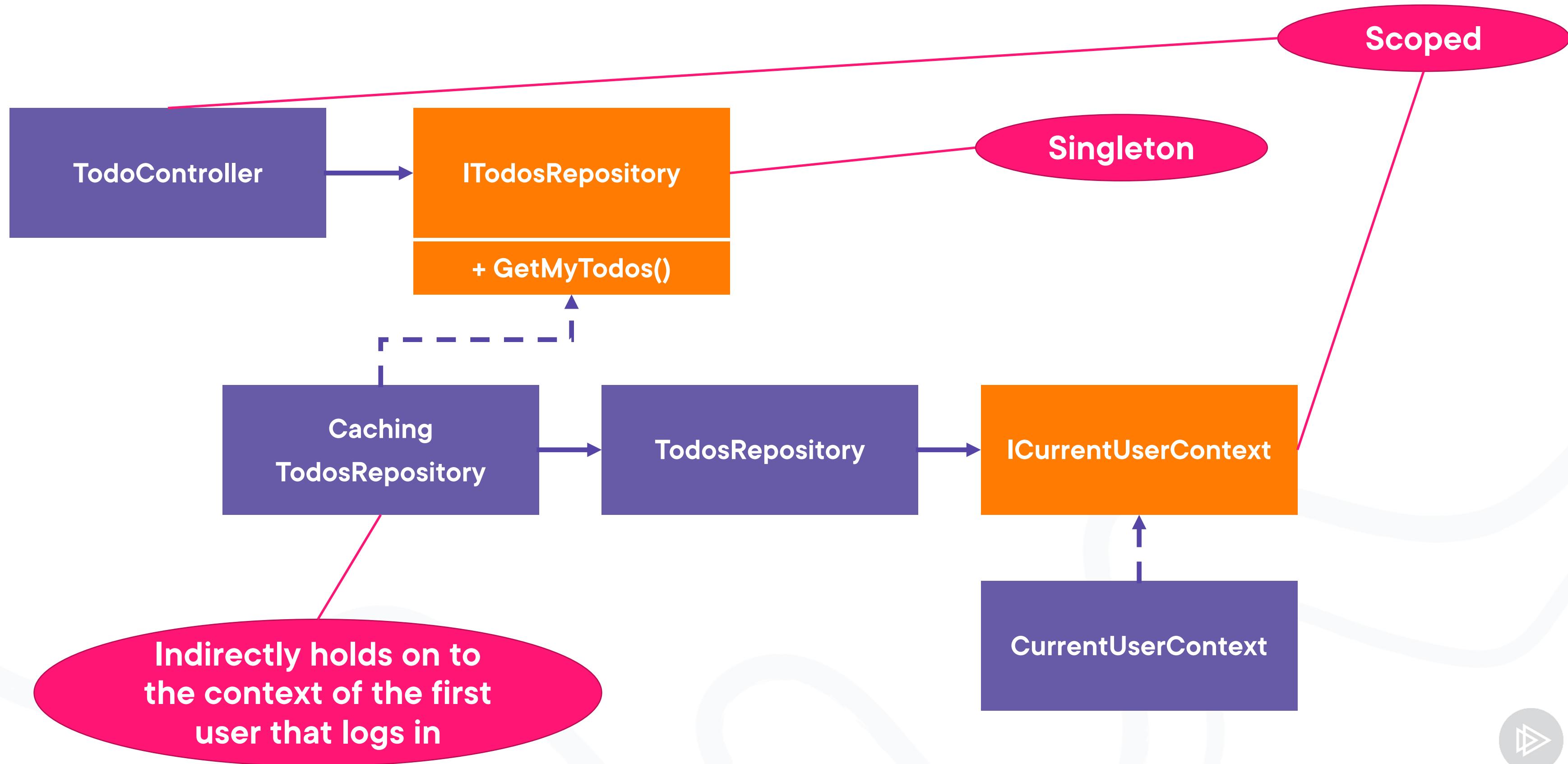


Hanging State Due to Lifetime Issues

A Repository Pattern Implementation



A Repository Pattern Implementation



Hanging state is often caused by dependency captivity



Demo



Dealing with dependencies that are expensive to instantiate



Demo

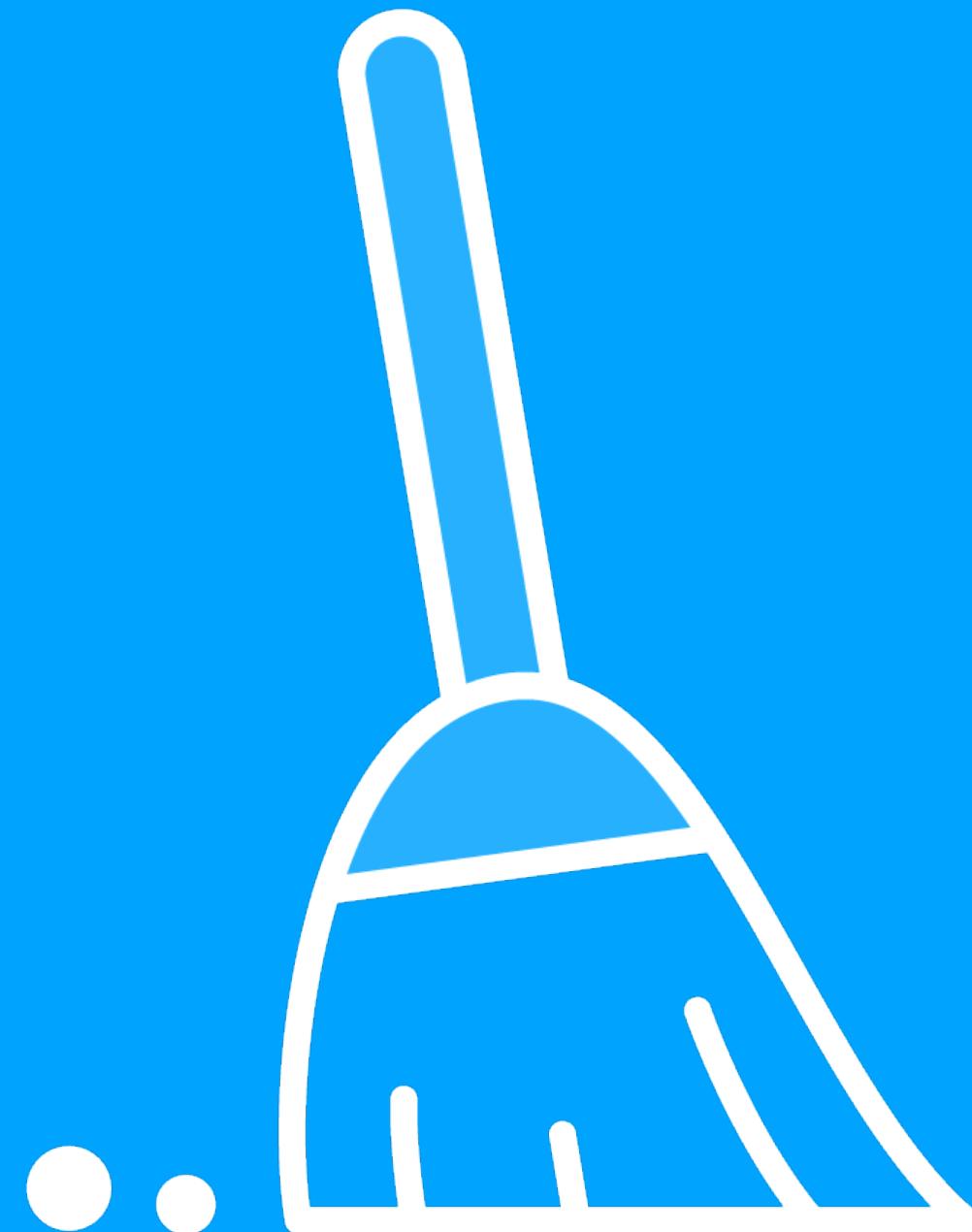


**Dependency constructor parameters that
are known only after constructing the
depending class**



| Dependency Injection and IDisposable





Who creates, also disposes

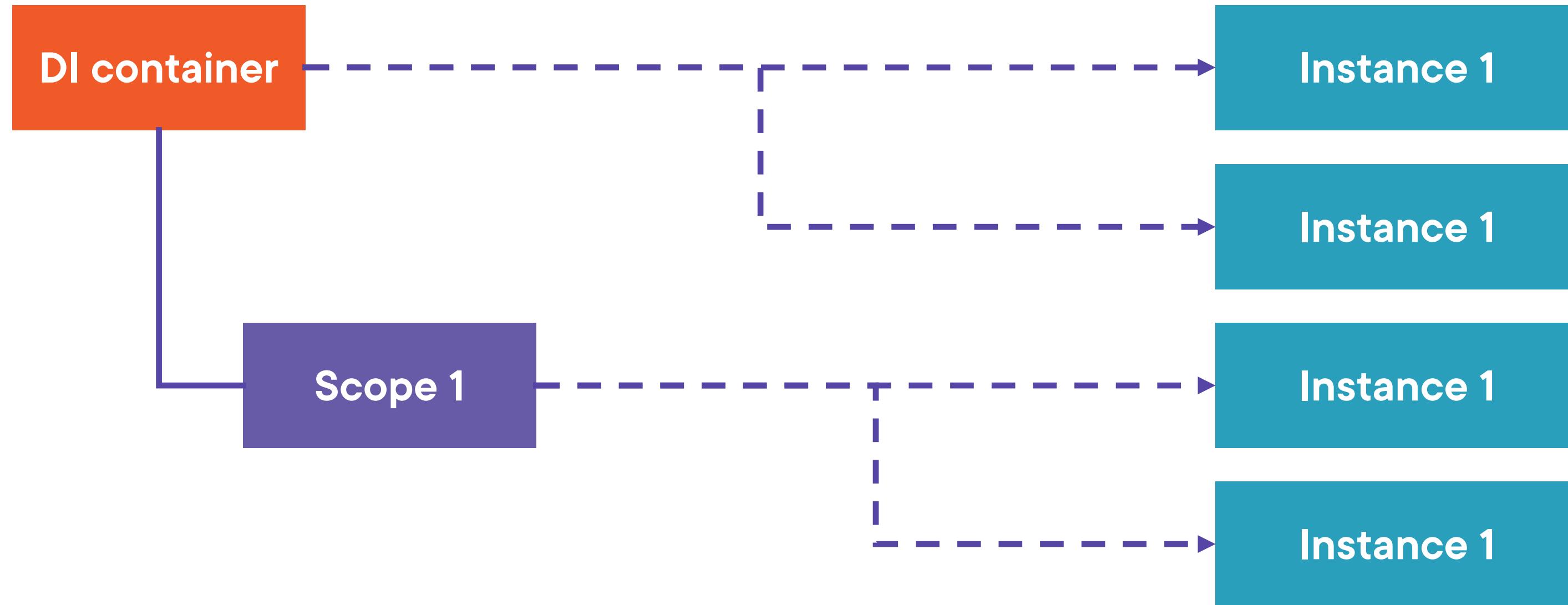
The DI container creates types on demand, so the DI container is also responsible for disposing them



All created types are
disposed when the
container or scope is
disposed



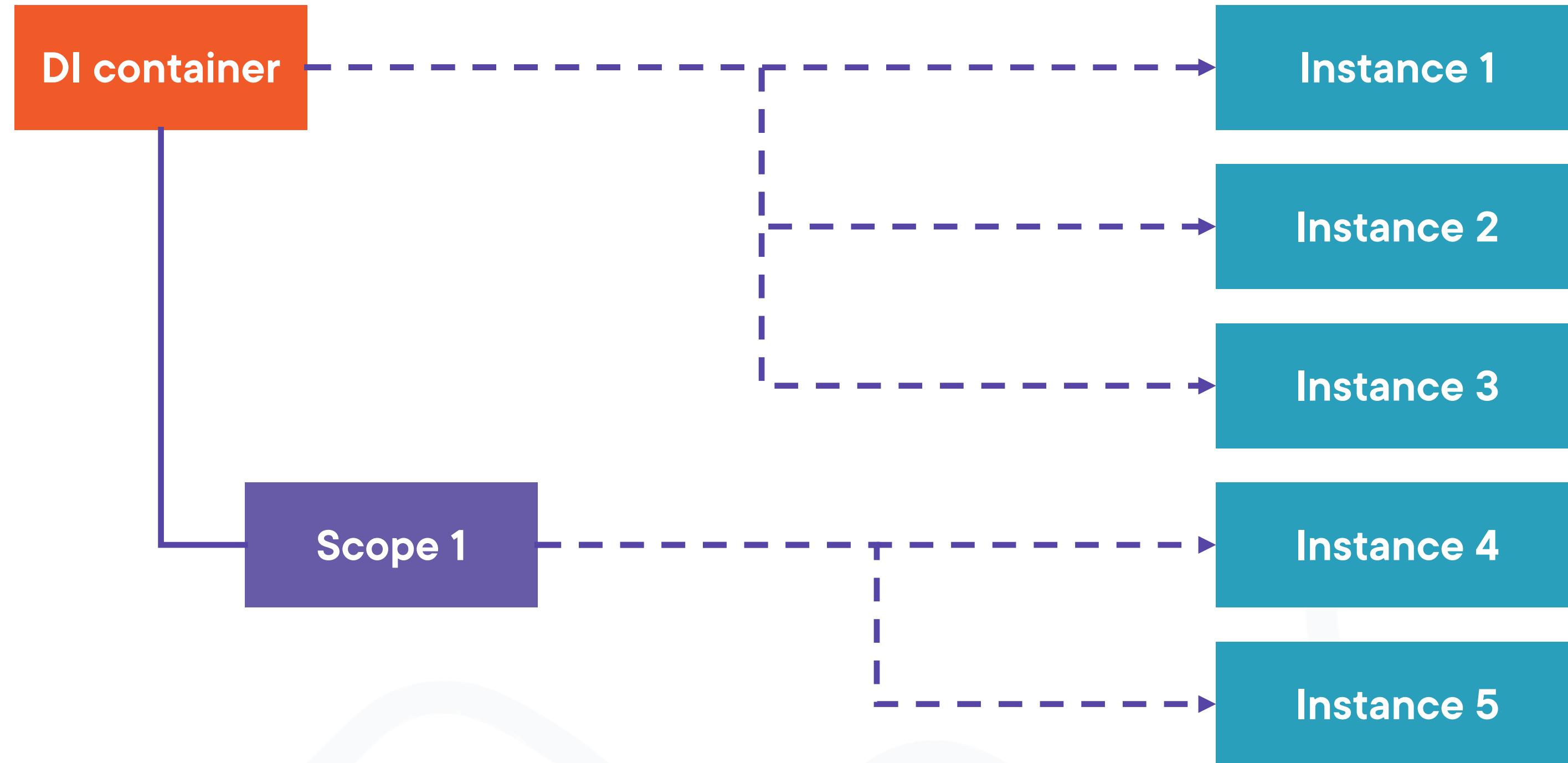
IDisposable and Singleton



**All good! Only one instance, and it
is disposed when the container is
disposed**



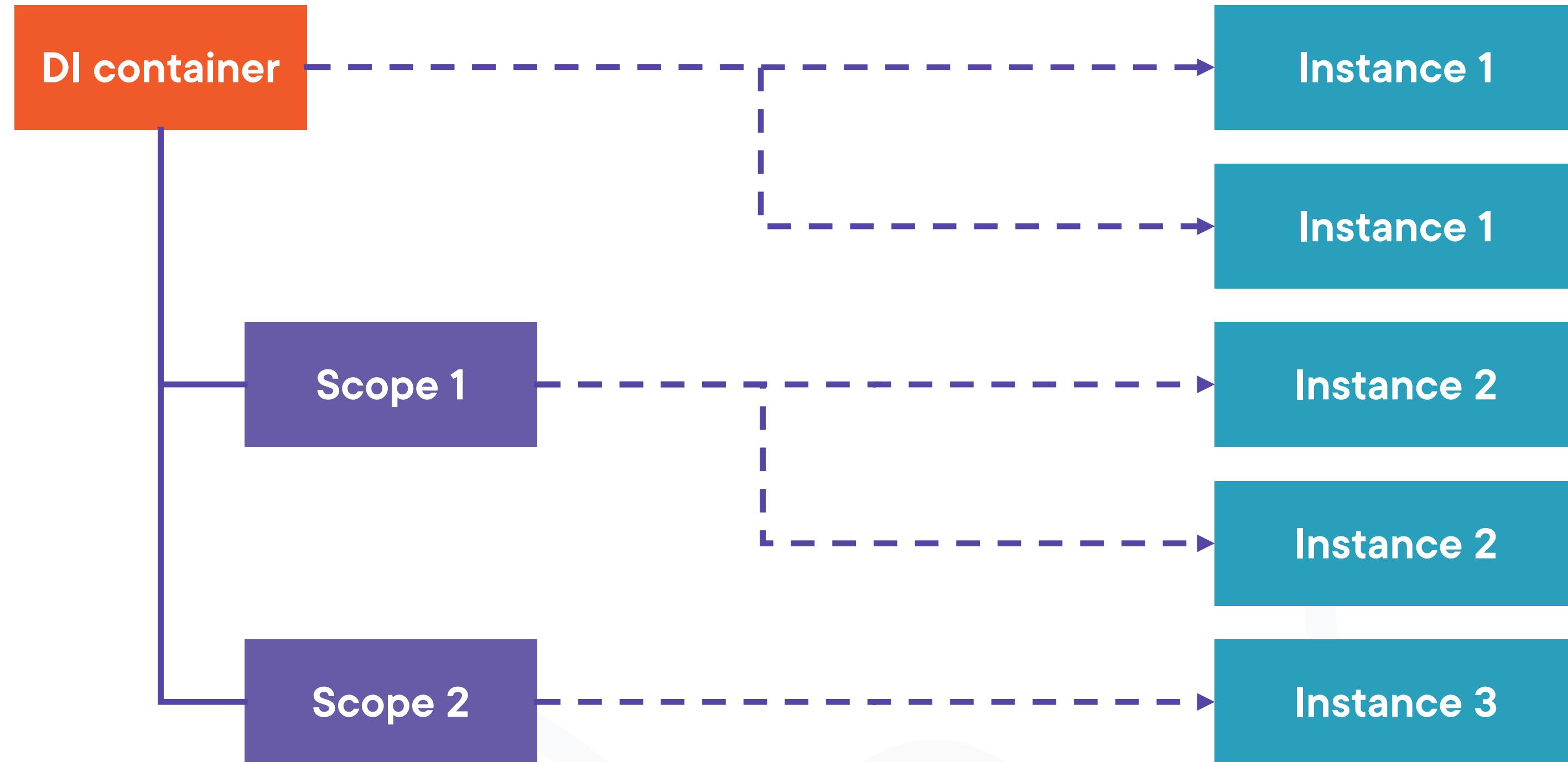
IDisposable and Transient



**More and more transient types
are created, but disposed only
much later when the container or
scope is disposed**



IDisposable and Scoped



**Scoped lifetimes resolved on a
scope are disposed when the
scope is disposed**



The instance associated with the root container is never disposed



Dos and Don't's



Do implement `IDisposable` in your classes as needed

Don't register `IDisposable` types as transient, instead instantiate them using a factory

Don't resolve `IDisposable` types in the root container

Avoid (custom) scopes and scoped `IDisposables` when you can



Demo



Dependency Injection and IDisposable



Overview



Implementing multiple service types

Hanging state

Deferring construction

**Working with constructor parameters that
can't be injected**

IDisposable: Dos and Don'ts



Up Next:

Advanced Techniques

