Beyond the Built-in Container

Steve Gordon
.NET Engineer and Microsoft MVP

@stevejgordon www.stevejgordon.co.uk

Overview

Extending the container using Scrutor

Scanning for services

Decorating services

Using a third-party container

Introducing and Installing Scrutor

Demo

Assembly scanning with Scrutor

- Benefits
- Reduce manual service registrations

Advantages of Assembly Scanning

As complexity grows, manual registration can be forgotten and go unnoticed

Assembly scanning ensures that new implementations are registered automatically

- Reduces risk
- Avoids manual code maintenance

Introduces a small, but largely negligible, startup performance impact

Considerations

Using marker interfaces leaks some implementation details

- Changes to dependencies may require changes to the marker interface to avoid invalid behavior

Consider use of marker interfaces carefully

- Prefer pragmatic simpler designs
- Profiling and real-world usage should drive a decision to apply more complex designs

Applying the Decorator Pattern with Scrutor

Decorator Pattern

Wrap functionality of services with additional functionality from an implementation sharing the same interface.

Demo

Apply the decorator pattern with Scrutor

- Apply caching to the weather forecaster

Decorators

Registered using the same lifetime as existing service descriptors

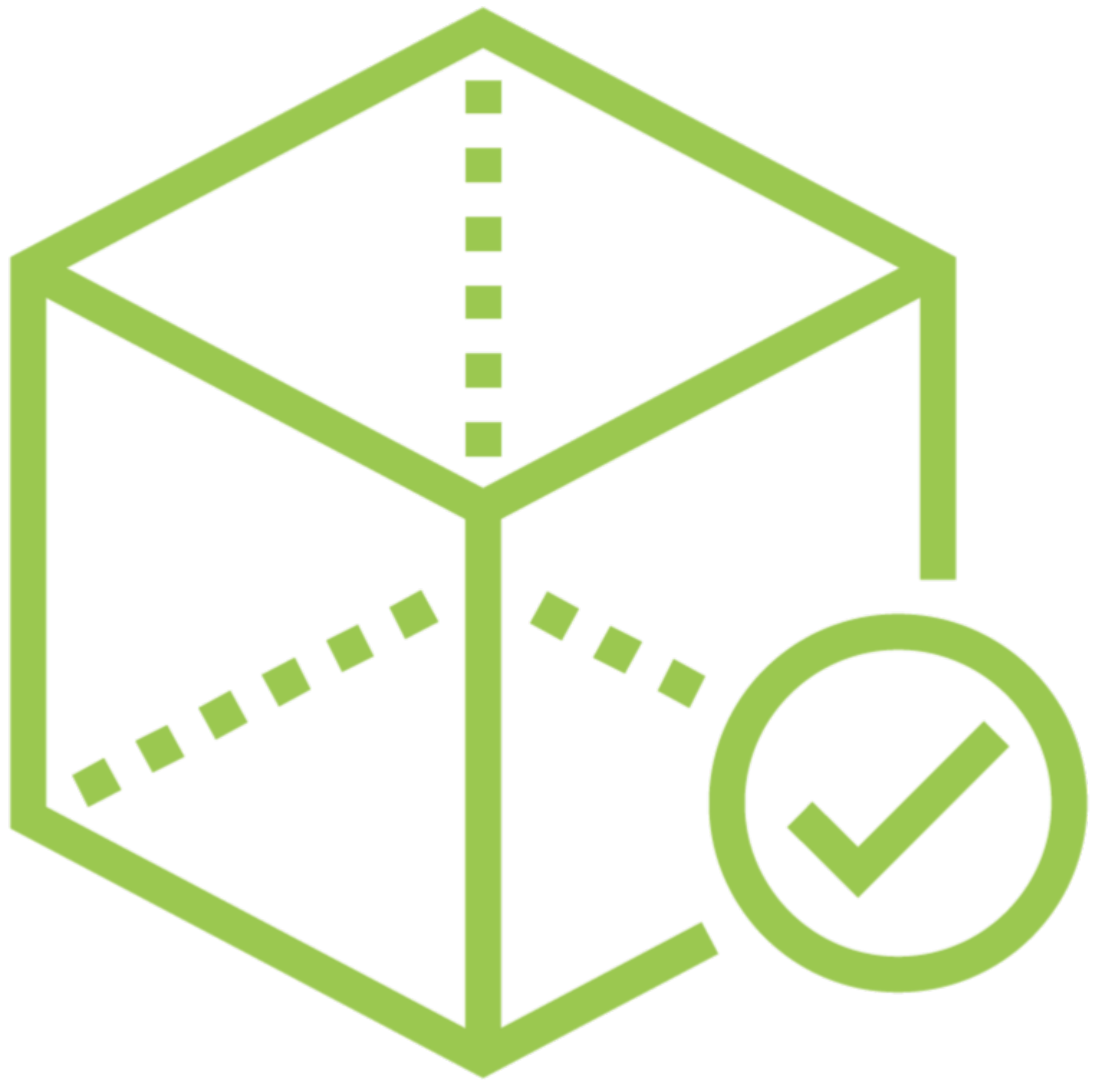
- The CachedWeatherForecaster “inherits” a singleton lifetime from the service it wraps

Decorators are a powerful way to extend applications without modifying existing code

- Scrutor’s implementation is usually sufficient

Third-party Conforming Containers

Conforming Containers



Many .NET dependency injection containers existed before the Microsoft DI container

- Design, features and implementation vary

Microsoft provide common abstractions

- Define the concept of conforming containers

Several third-party containers conform to the Microsoft abstraction

Third-party Containers

Autofac

Dryloc

Grace

LightInject

Lamar

Stashbox

Unity

Simple Injector

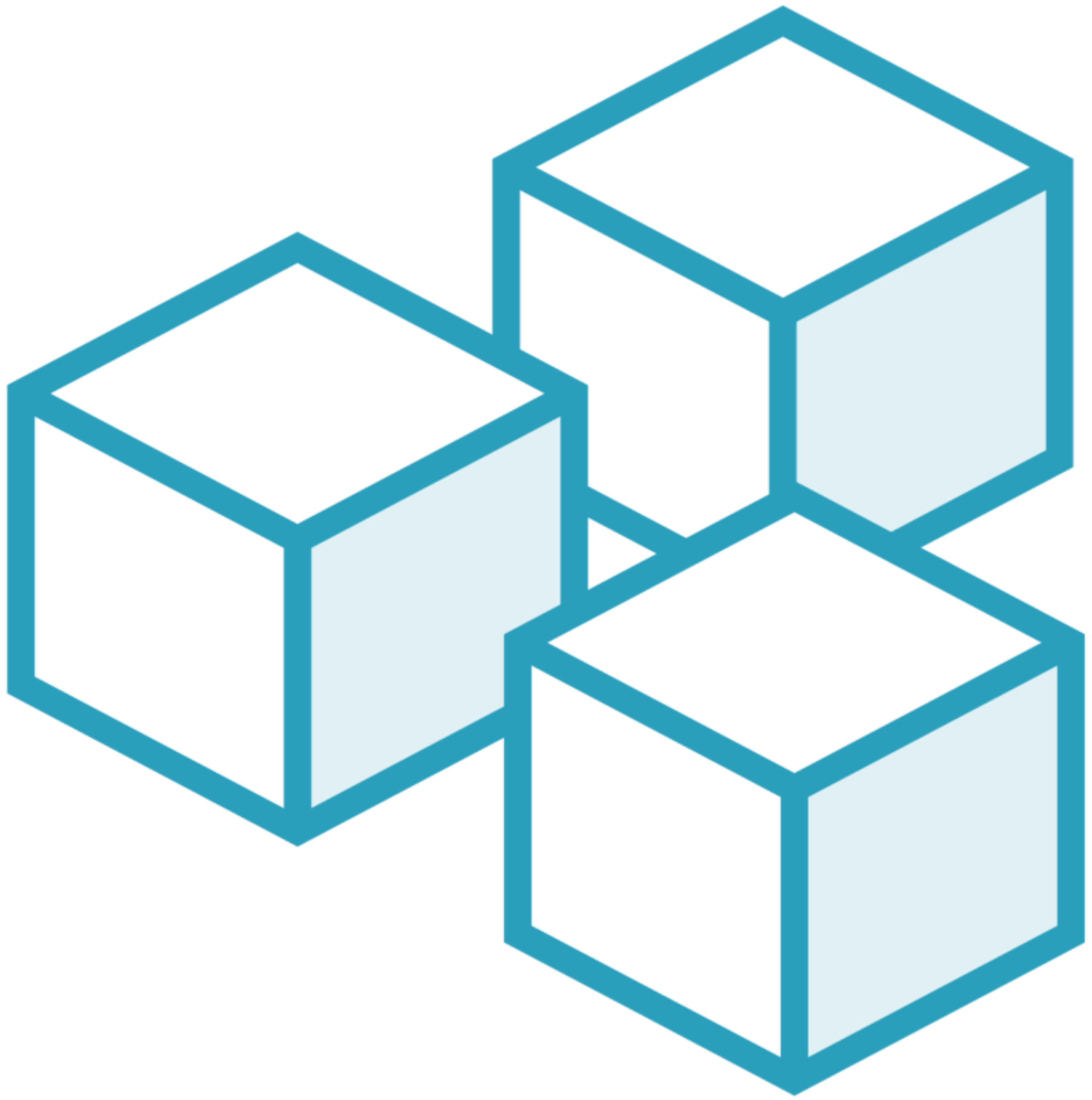
Conforming Containers

Many third-party containers offer additional advanced features

Microsoft recommends using their container as the default choice

Evaluate your choice of container carefully

Can be useful when migrating legacy applications to ASP.NET Core



Demo

Install and utilize Autofac

- Configure ASP.NET Core to use Autofac instead of the built-in container
- Learn how the containers coordinate
- Migrate registrations to the Autofac APIs

Do We Required a Third-party Container?

The Microsoft container is feature limited by-design

We can easily swap in a more feature-rich third-party container

This demo doesn't really justify a need for Autofac

- Adding Autofac is shown as an example

For most applications, their demands will be met by the default Microsoft container

Review

Congratulations!

Covered core concepts of the Microsoft dependency injection container

Explored advanced application features requiring more complex registrations

Return to specific modules to recap the techniques they demonstrate

Steve Gordon

MICROSOFT DEVELOPER TECHNOLOGIES MVP

@stevejgordon www.stevejgordon.co.uk