Advanced Techniques

Henry Been
Independent DevOps & Azure Architect

@henry_been www.henrybeen.nl

Overview

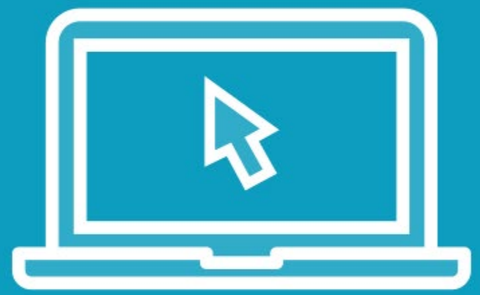
Using the DI container for A/B testing

Switching to another DI container

Autofac specific techniques:

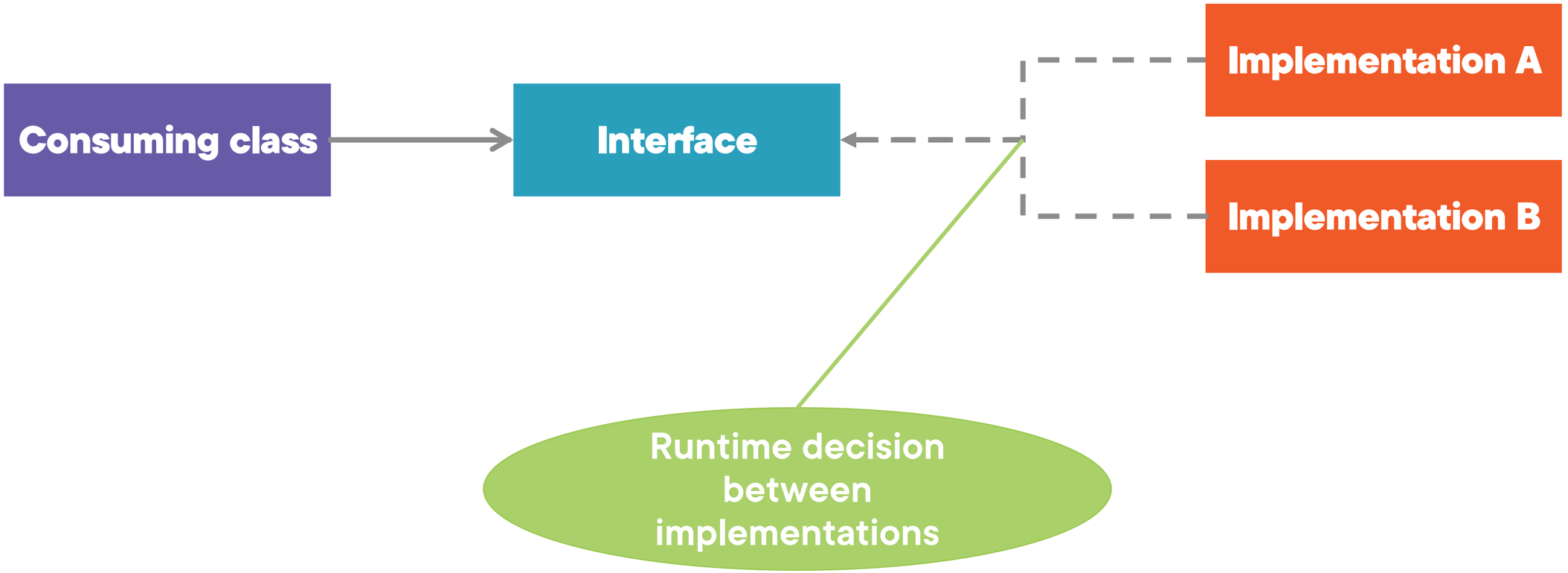
- Property injection
- Built-in support for Lazy<T>
- Interception
- Assembly scanning

Demo

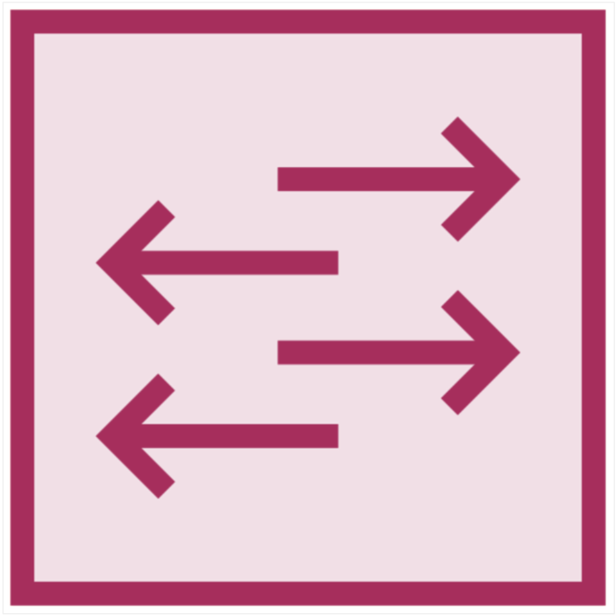


Implementing A/B testing using the DI Container

A/B Testing


Consuming class
Interface
Implementation A
Implementation B
Runtime decision between implementations

Choosing Another Container



Why Switch Containers?

Historical reasons

Specific capabilities

Claimed performance (actually not true)

Autofac vs. Ninject

Autofac

A very feature-complete container

Ninject

A container that focusses on simplicity

Autofac

About 140M downloads on NuGet

Very feature-complete

Built-in support for modules

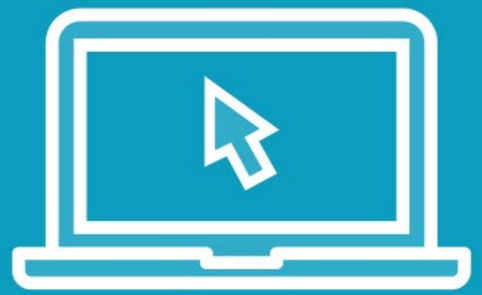
Ninject

About 27M downloads on NuGet

Focusses on simplicity

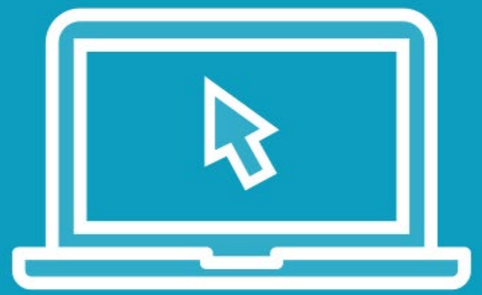
Ultra lightweight and portable

Demo



Switching to Autofac

Demo



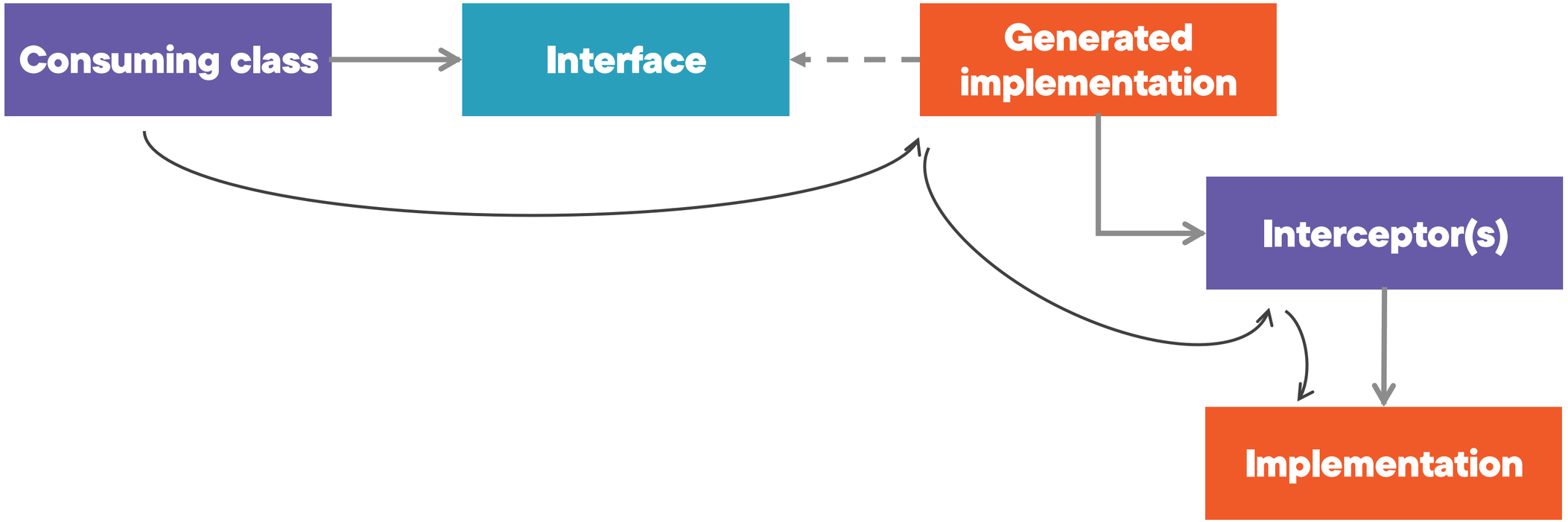
Optional and missing dependencies

- Using Lazy<T>
- Using = null
- Using property injection

Interception

Adding Interception

Adding Interception



Common Use Cases

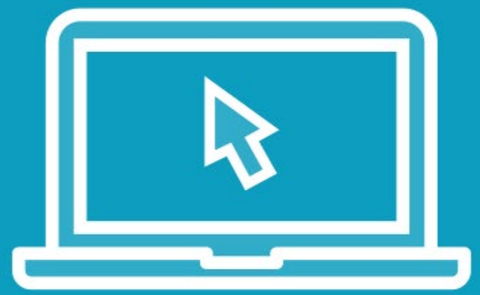
Logging

To quickly add rudimentary logging to an application that doesn't have it

Exception hiding

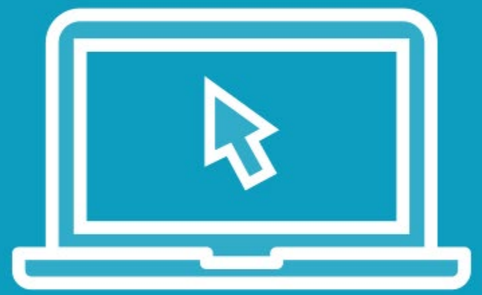
Adding a layer to catch and log exceptions, and to strip stack traces to prevent information leakage

Demo



Using Autofac interception for logging calls

Demo



Using Autofac assembly scanning to automatically register dependencies

Summary

- A/B Testing
- Switching to Autofac
- Property injection
- Support for Lazy<T>
- Interception
- Assembly scanning

What's Next?

More Information

Dependency Injection in ASP.NET Core 6

Steve Gordon

C# 10 Learning Path

To continue your journey learning C#

Congratulations and thank you!