Working with Generic Interfaces

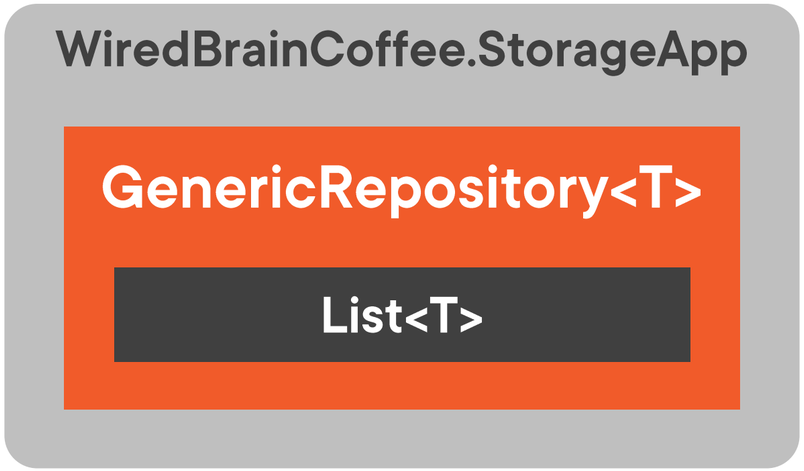
Thomas Claudius Huber
Software Developer

@thomasclaudiush www.thomasclaudiushuber.com

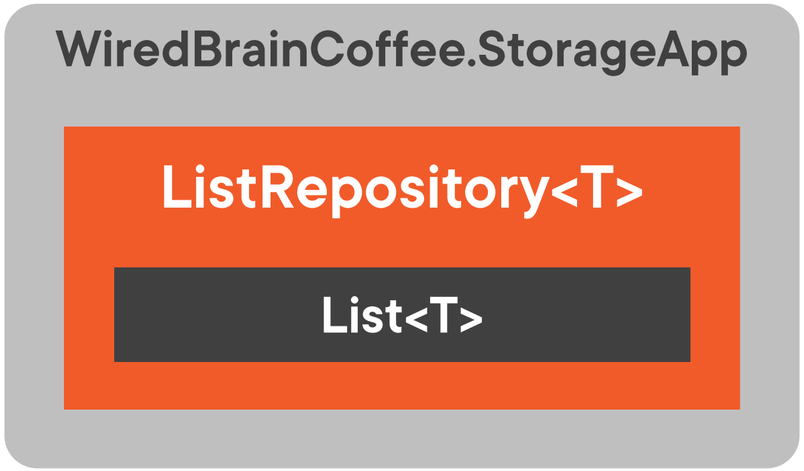
Module Outline

- Why use a generic interface?
- Create a generic interface
  - Use a generic interface
  - Understand covariance
  - Understand contravariance
- Work with interface inheritance

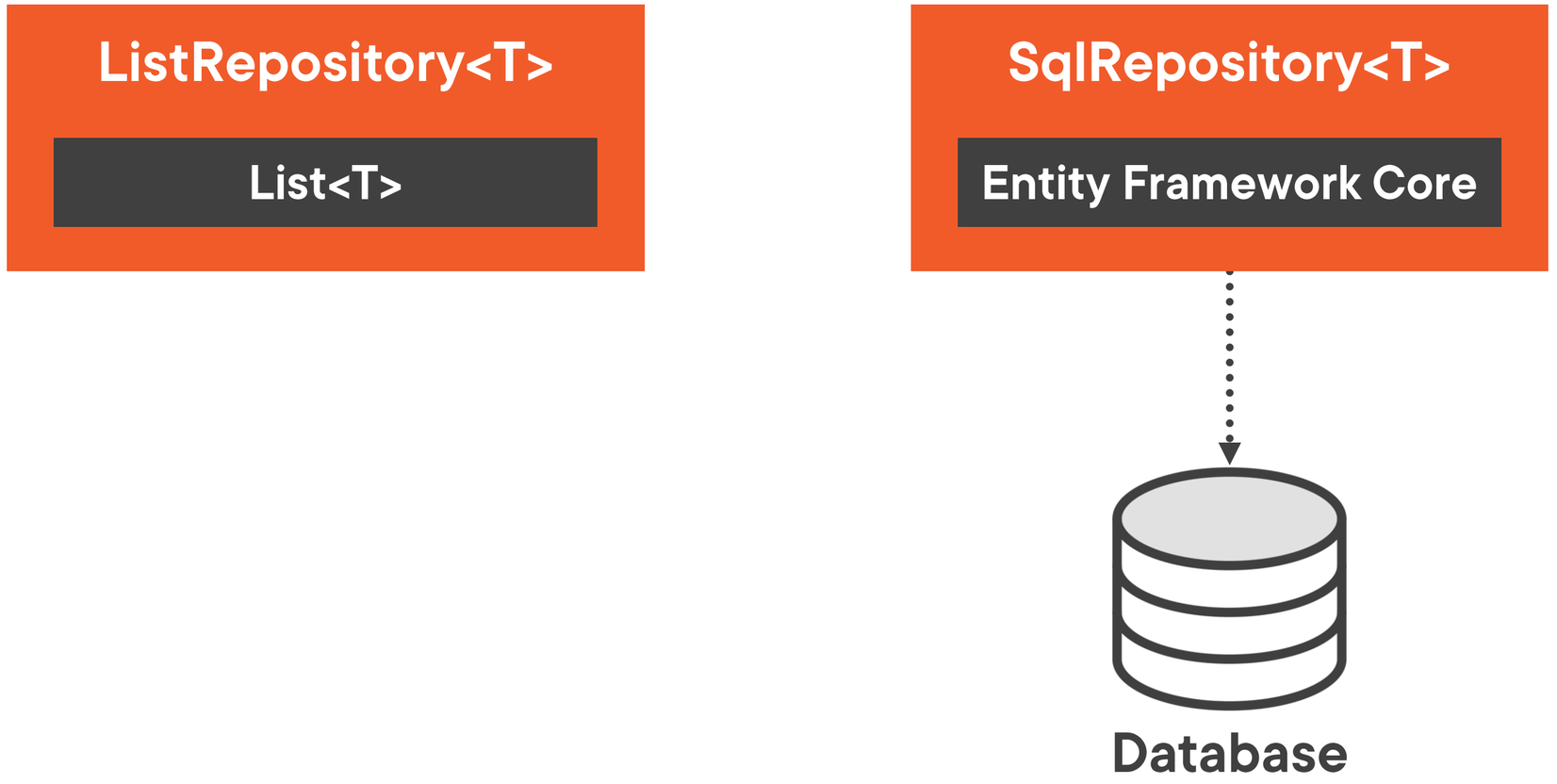
Why Use a Generic Interface?



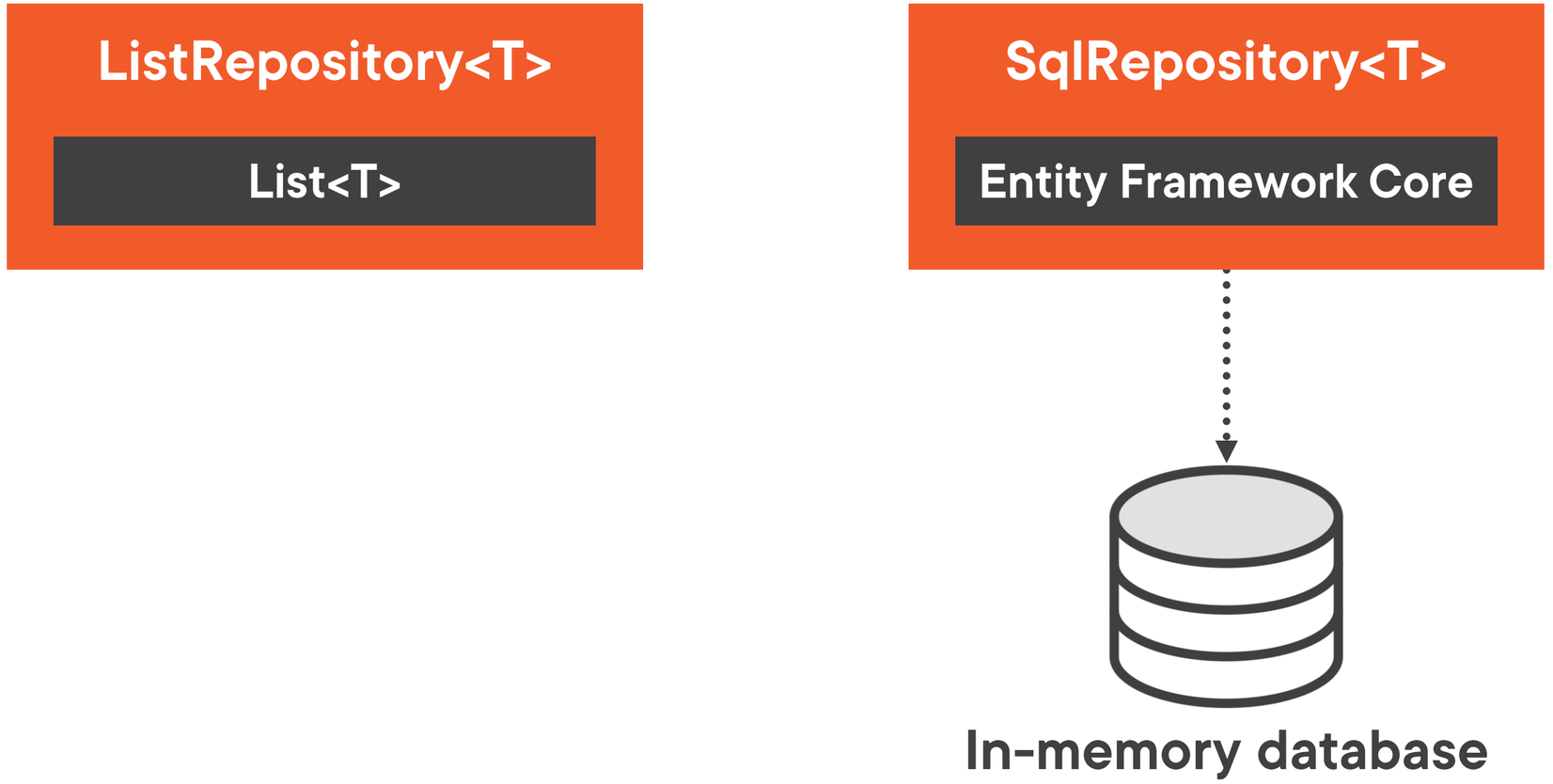
Why Use a Generic Interface?



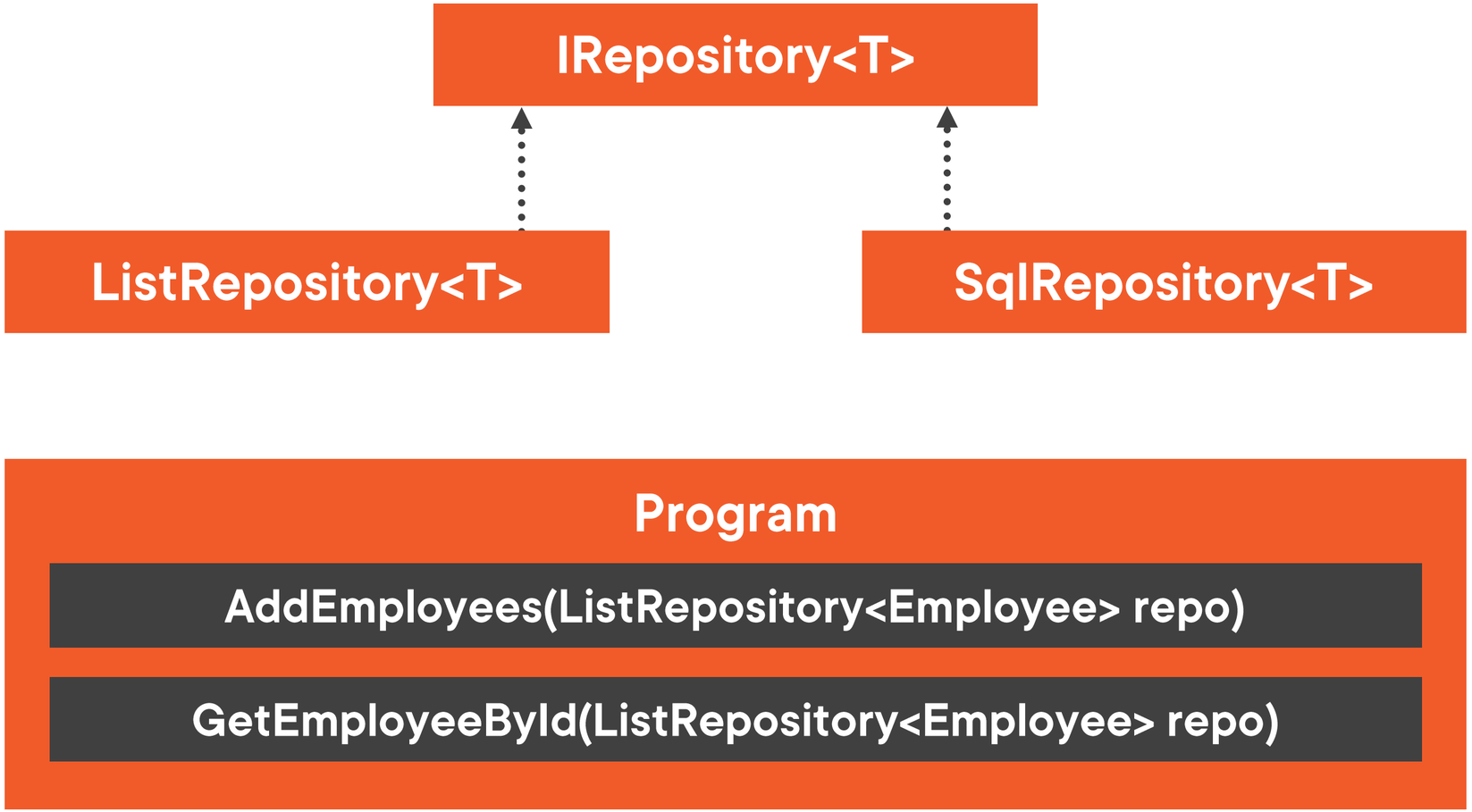
Why Use a Generic Interface?



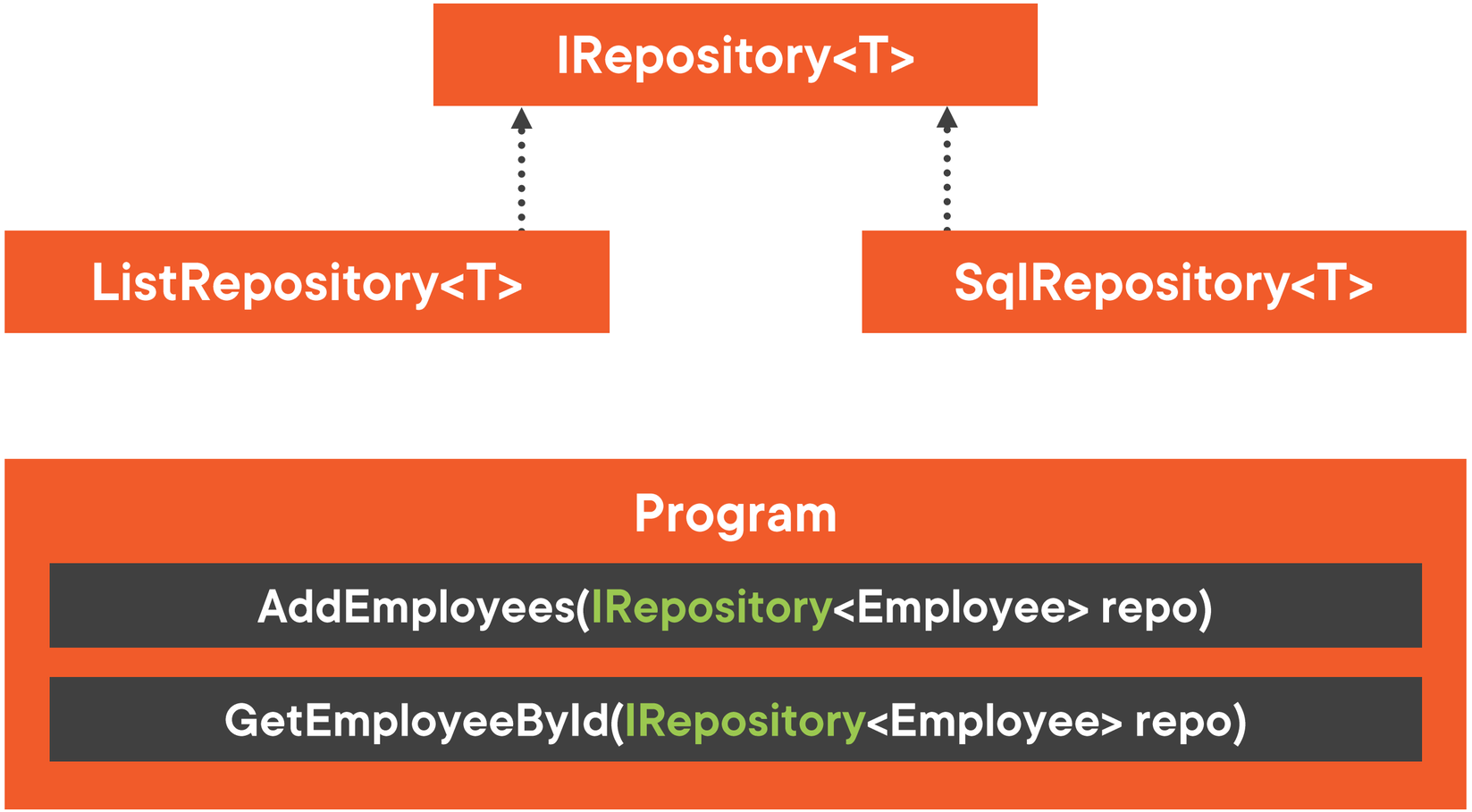
Why Use a Generic Interface?



Why Use a Generic Interface?



Why Use a Generic Interface?



Dependency Inversion Principle

Components must depend on abstractions and not on implementations

Demo

Build a SqlRespository<T> class

- **Use Entity Framework Core**
- **Store data in an in-memory database**

Demo

Create a generic interface

Use an Existing Generic Interface

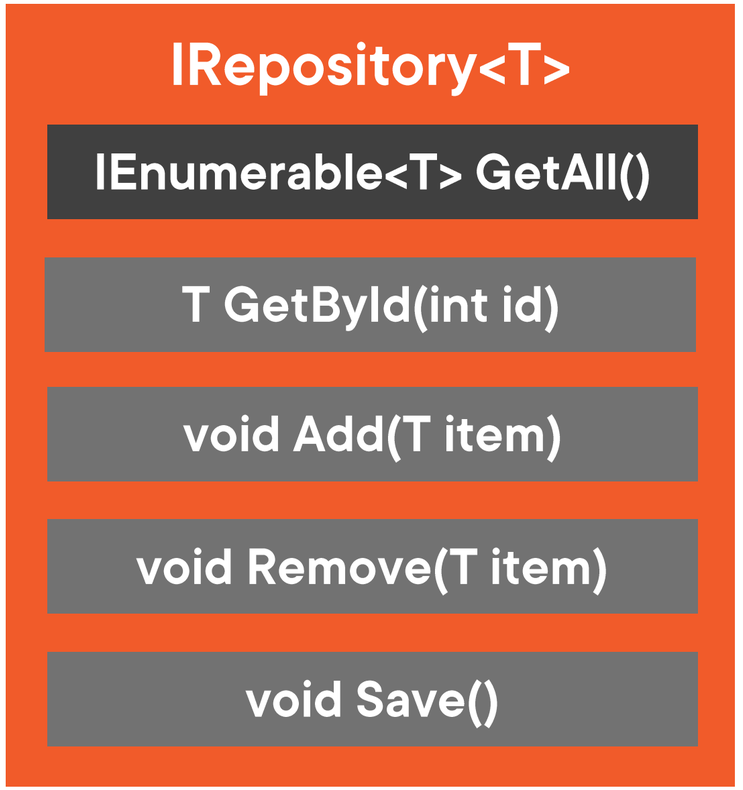
System.Collections.Generic

IEnumerable<T>

Iterable with a foreach loop

Implemented by
- Generic collection classes
- All C# arrays

LINQ methods are extension methods of this interface



Demo

Add and implement the GetAll method

Use the existing IEnumerable<T> interface

Demo

Understand covariance

Demo

Understand contravariance

Demo

Work with interface inheritance

Summary

- Create and use a generic interface
- Generic type parameters are invariant
- Declare type parameter as covariant
  - <out T> - works if T is used only for return values
  - Use a less specific generic type argument on the generic interface
- Declare type parameter as contravariant
  - <in T> - works if T is used only for input parameters
  - Use a more specific generic type argument on the generic interface
- Work with interface inheritance

Up Next:
Creating Generic Methods and Delegates