DI Why?

Getting a Grip on Dependency Injection

Jeremy Clark
jeremybytes.com
github.com/jeremybytes

Typical Introduction

```
private void BuildMainWindow()
{
    var builder = new ContainerBuilder();
    builder.RegisterType<SQLReader>().As<IPersonReader>()
        .SingleInstance();
    builder.RegisterSource(
        new AnyConcreteTypeNotAlreadyRegisteredSource());
    IContainer Container = builder.Build();
    Application.Current.MainWindow =
        Container.Resolve<PeopleViewerWindow>();
}
```

What Is Dependency Injection?

- Dependency Injection is a software design pattern that allows a choice of component to be made at run-time rather than compile time.
  - Wikipedia 2012

What Is Dependency Injection?

- Dependency injection is a software design pattern that allows the removal of hard-coded dependencies and makes it possible to change them, whether at run-time or compile-time.
  - Wikipedia 2013

What Is Dependency Injection?

- Dependency injection is a software design pattern that implements inversion of control and allows a program design to follow the dependency inversion principle. The term was coined by Martin Fowler.
  - Wikipedia 2014

What Is Dependency Injection?

- In software engineering, dependency injection is a software design pattern that implements inversion of control for software libraries, where the caller delegates to an external framework the control flow of discovering and importing a service or software module. Dependency injection allows a program design to follow the dependency inversion principle where modules are loosely coupled. With dependency injection, the client part of a program which uses a module or service doesn't need to know all its details, and typically the module can be replaced by another one of similar characteristics without altering the client.

  - Wikipedia 2015

What Is Dependency Injection?

- In software engineering, dependency injection is a software design pattern that implements inversion of control for resolving dependencies. A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it. The service is made part of the client's state.[1] Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern.

  - Wikipedia 2016

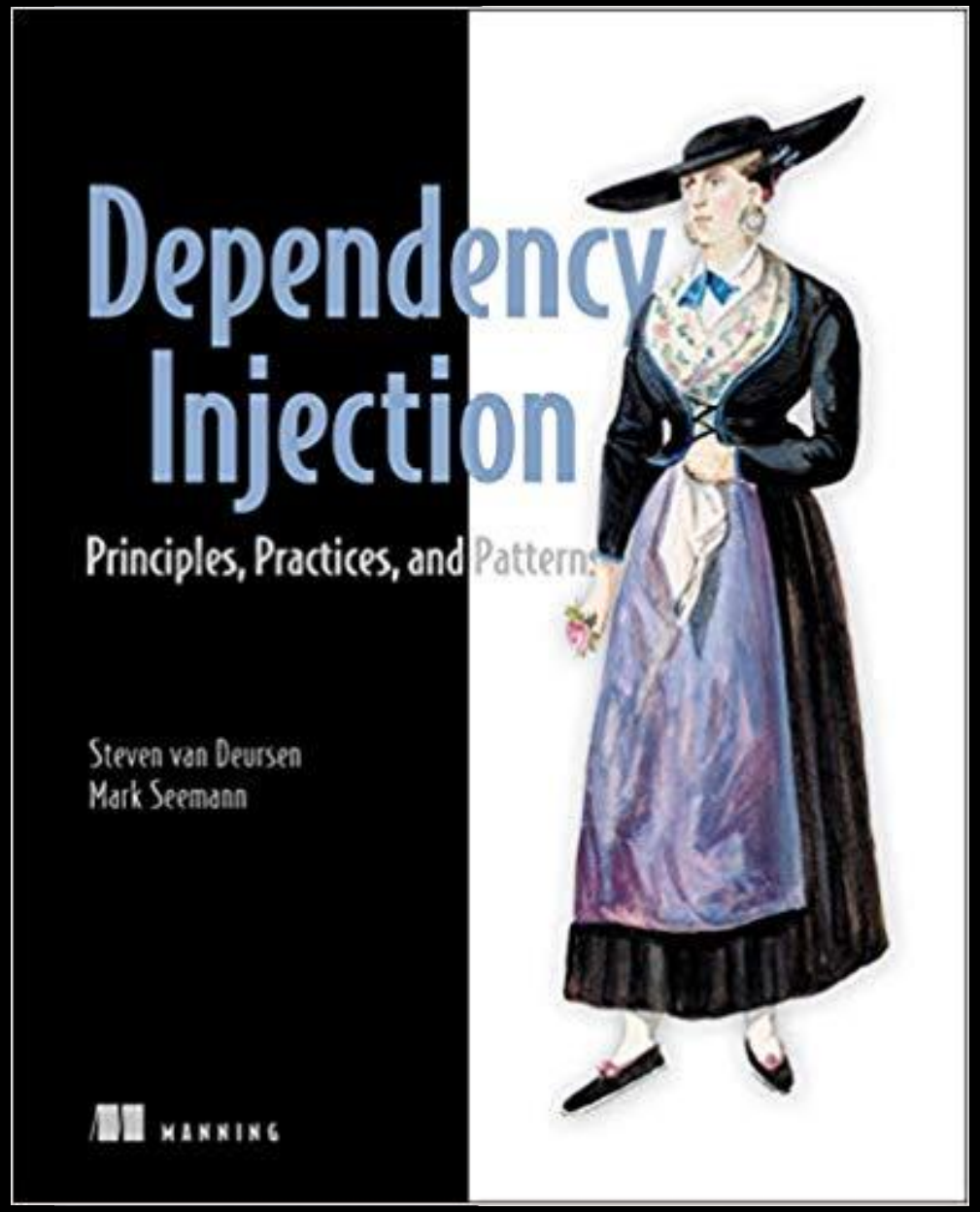
What Is Dependency Injection?

- Dependency Injection is a set of software design principles and patterns that enable us to develop loosely coupled code.
  - Mark Seemann

Dependency Injection

Principles, Practices, and Patterns

- Mark Seemann
- Steven van Deursen



Primary Benefits

- Extensibility
- Parallel Development
- Maintainability
- Testability
- Late Binding

- Adherence to S.O.L.I.D. Design Principles.

Extensibility

Code can be extended in ways not explicitly planned for.

Parallel Development

Code can be developed in parallel with less chance of merge conflicts.

Maintainability

Classes with clearly defined responsibilities are easier to maintain.

Testability

Classes can be unit tested,
i.e., easily isolated from other classes
and components for testing.

Late Binding

Services can be swapped with other services without recompiling code.

SOLID Principles

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

Primary Benefits

- Extensibility
- Parallel Development
- Maintainability
- Testability
- Late Binding

- Adherence to S.O.L.I.D. Design Principles.

Dependency Injection Concepts

- DI Design Patterns
  - Constructor Injection
  - Property Injection
  - Method Injection
  - Ambient Context
  - Service Locator
- Dimensions of DI
  - Object Composition
  - Interception
  - Lifetime Management

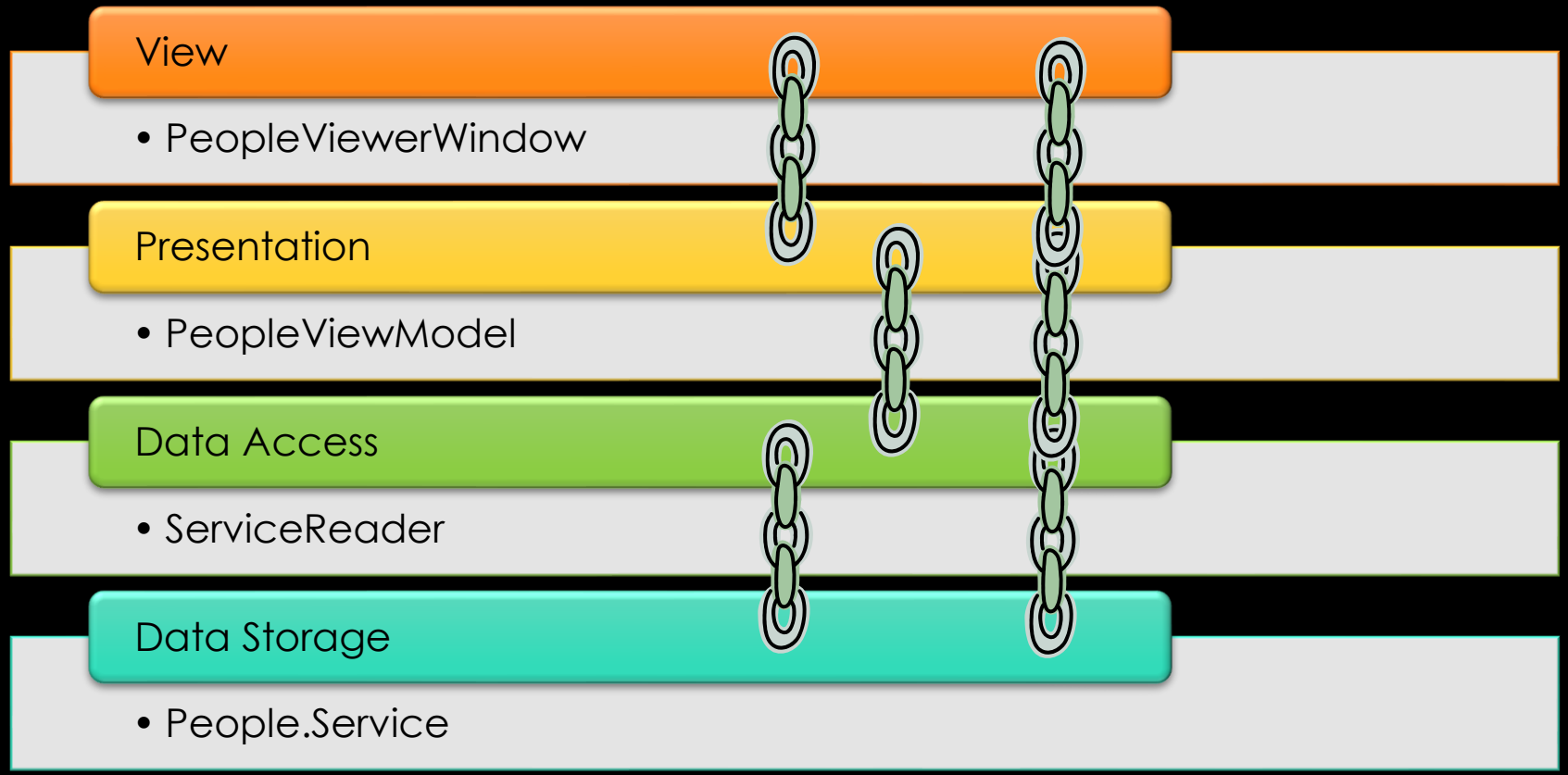
Dependency Injection Containers

- C# Containers
  - Autofac
  - Ninject
- Frameworks w/ Containers
  - ASP.NET Core
  - Angular
  - Prism

and many others

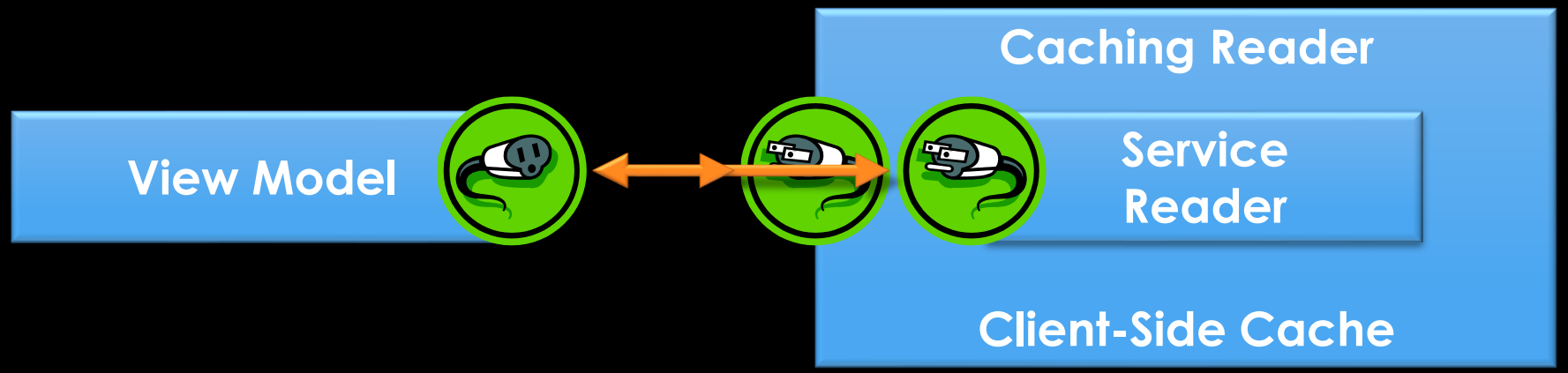
Application Layers

Tight Coupling

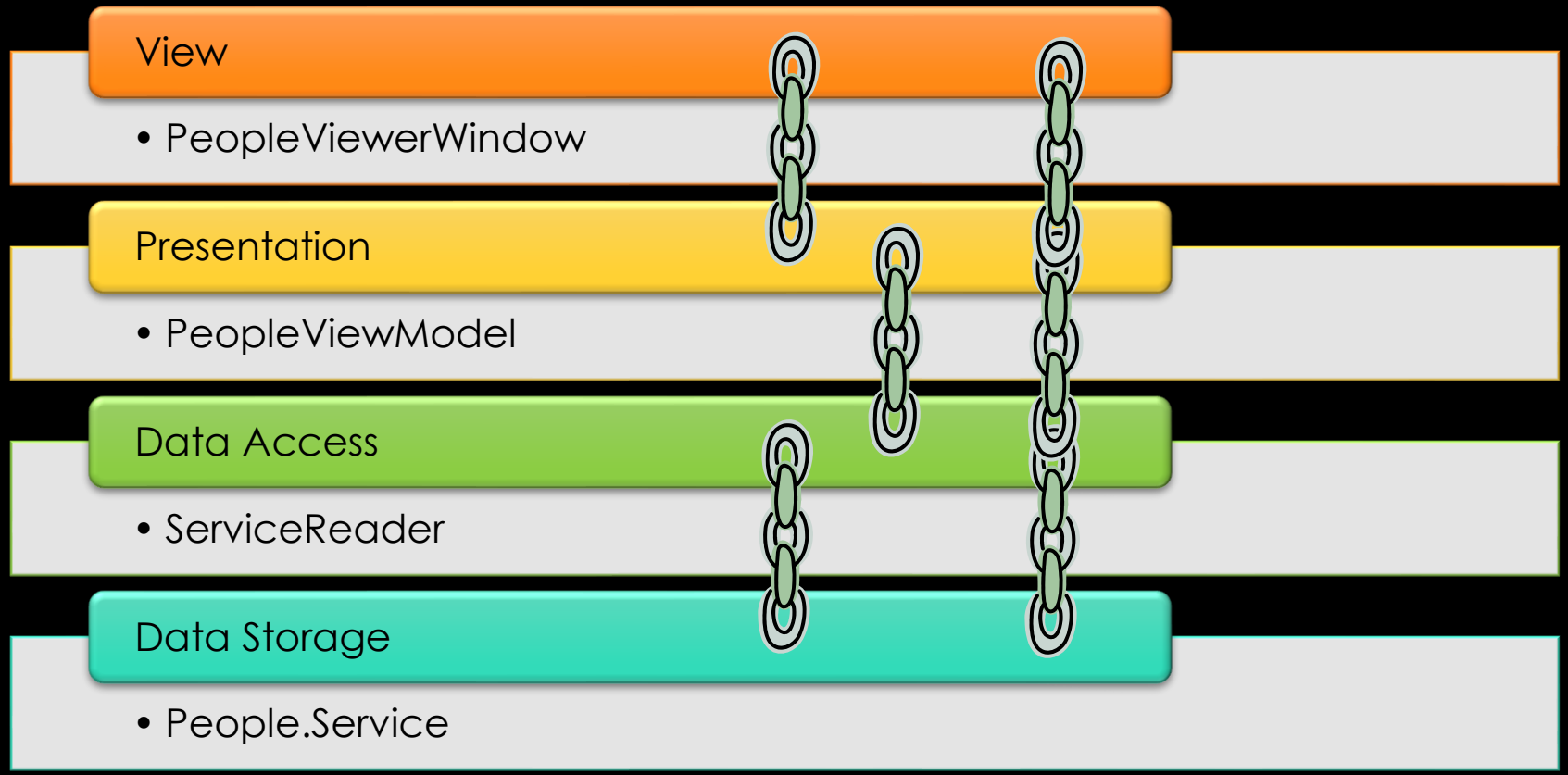


Creating a Caching Reader

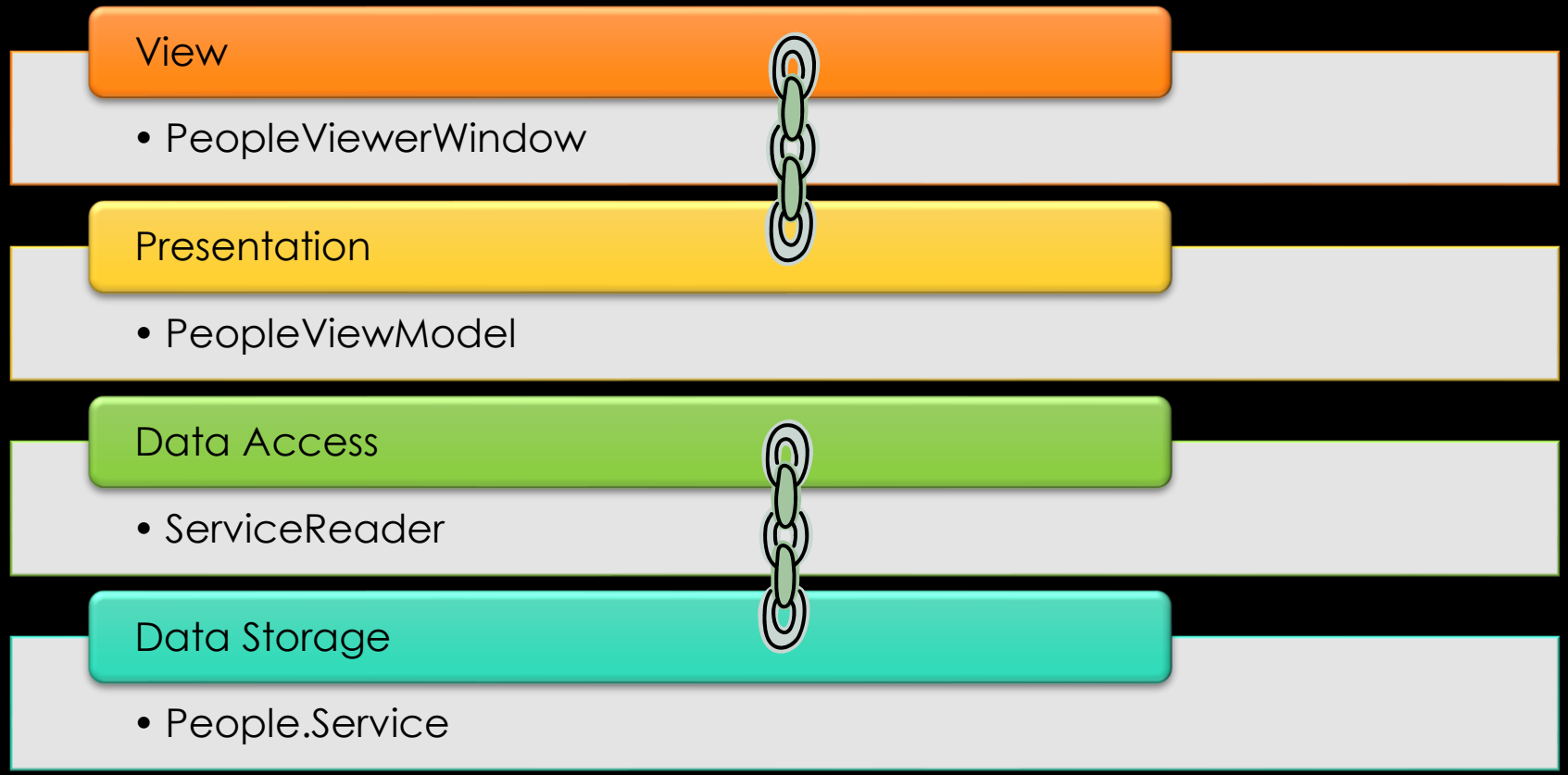
The Decorator Pattern



Loose(r) Coupling



Loose(r) Coupling



Primary Benefits

- Extensibility
- Parallel Development
- Maintainability
- Testability
- Late Binding

- Adherence to S.O.L.I.D. Design Principles.

Dependency Injection Concepts

- DI Design Patterns
  - Constructor Injection
  - Property Injection
  - Method Injection
  - Ambient Context
  - Service Locator
- Dimensions of DI
  - Object Composition
  - Interception
  - Lifetime Management

Thank You!

Jeremy Clark

- jeremybytes.com
- jeremy@jeremybytes.com
- github.com/jeremybytes

https://github.com/jeremybytes/dependency-injection-net9