

DI Why?

Getting a Grip on Dependency Injection

Jeremy Clark

www.jeremybytes.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.

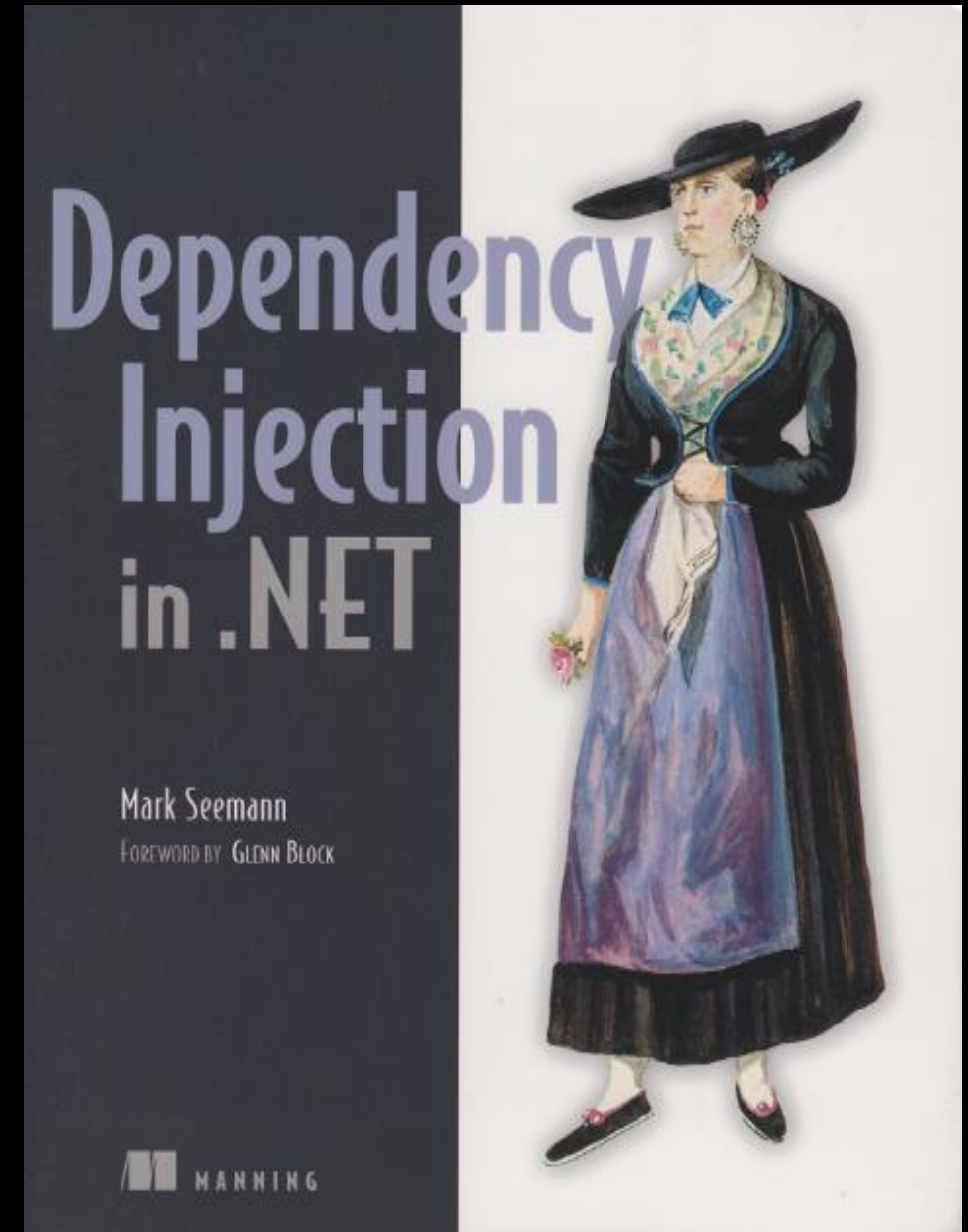
- Wikipedia 2016

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 in .NET

- Mark Seemann



Dependency Injection

Principles, Practices, and Patterns

- Steven von Deursen
- Mark Seemann

Dependency Injection

Principles, Practices, and Patterns

Steven van Deursen
Mark Seemann

 MANNING



Primary Benefits

- Extensibility*
 - Late Binding*
 - Parallel Development
 - Maintainability
 - Testability*
-
- Adherence to S.O.L.I.D. Design Principles.

*Topics we'll touch on today

Dependency Injection Concepts

- DI Design Patterns
 - Constructor Injection*
 - Property Injection*
 - Method Injection
 - Ambient Context
 - Service Locator

- DI Containers
 - Unity
 - Castle Windsor
 - Ninject*
 - Autofac*
 - Spring.NET
 - and others

- Object Composition*

*Topics we'll touch on today

Application Layers

View

- MainWindow

Presentation

- PeopleViewModel

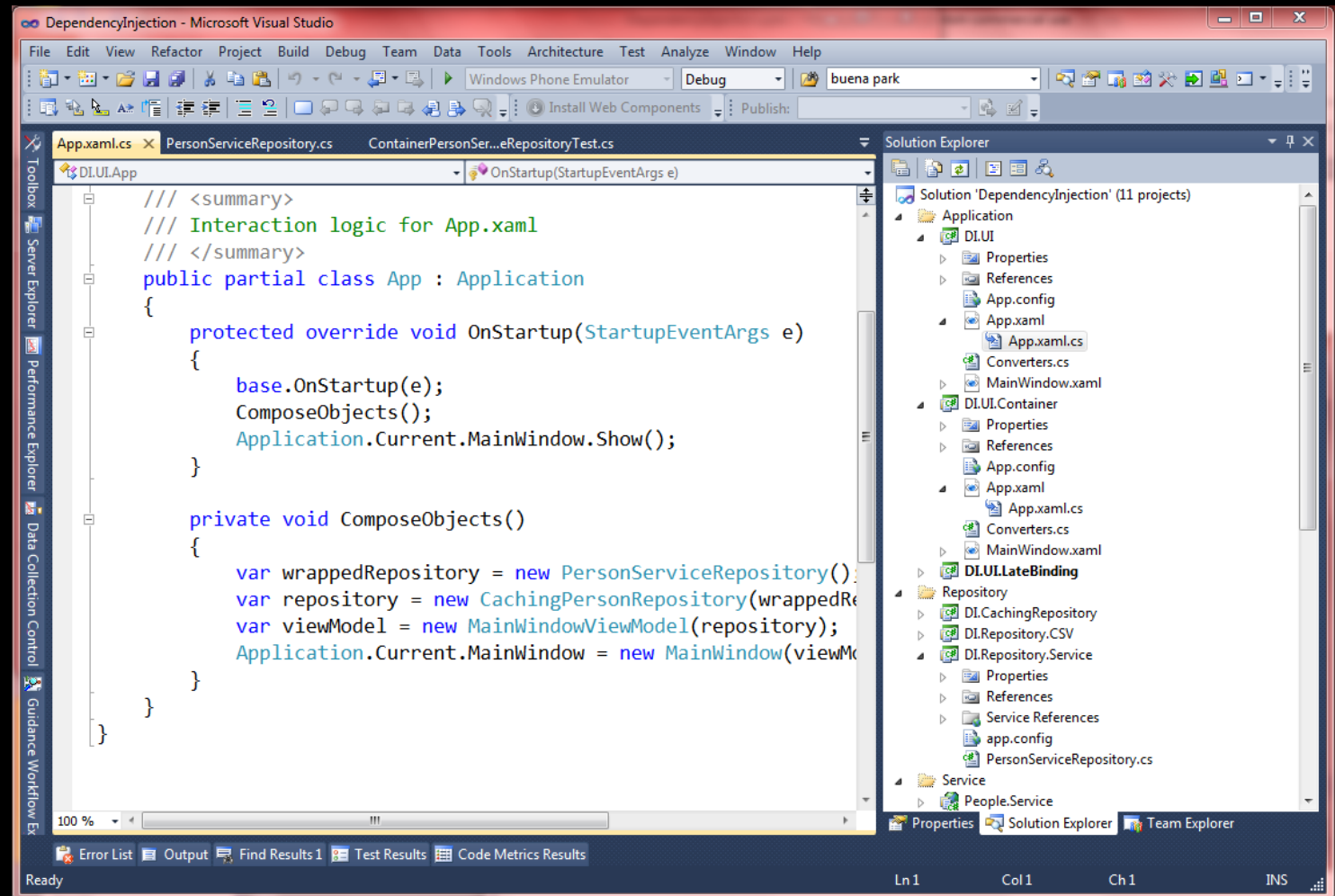
Data Access

- ServiceReader

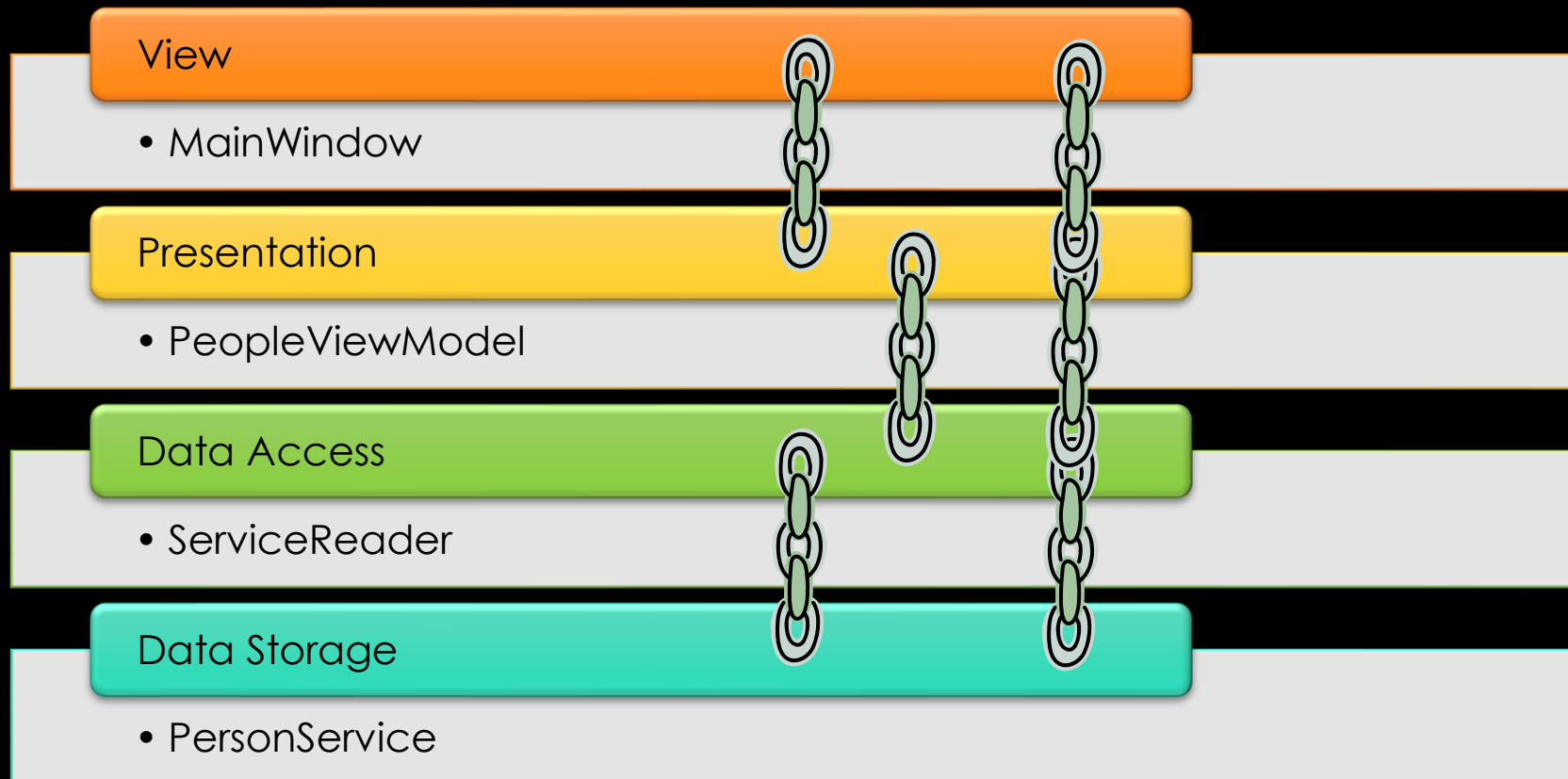
Data Storage

- PersonService

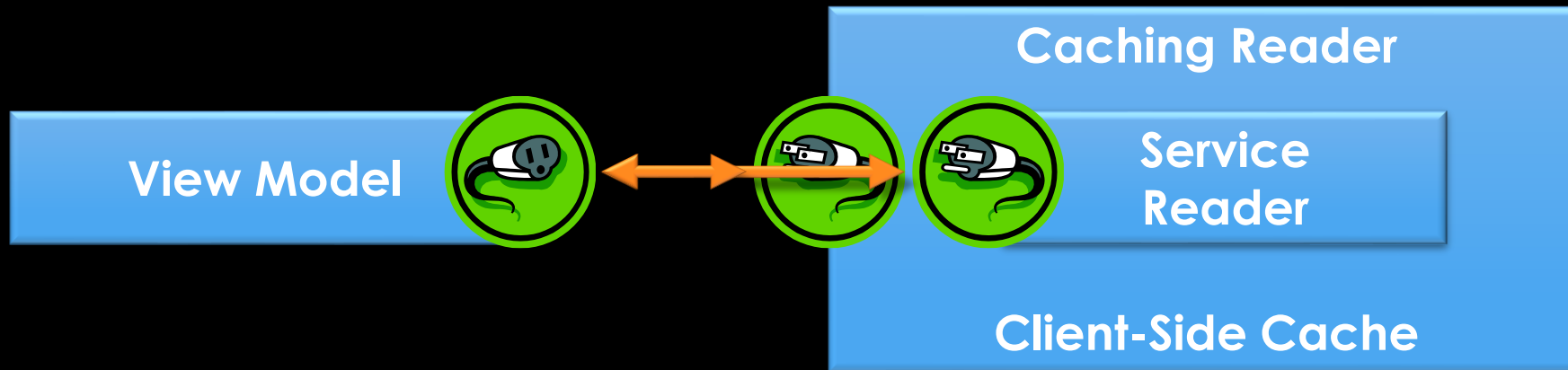
Look At The Code



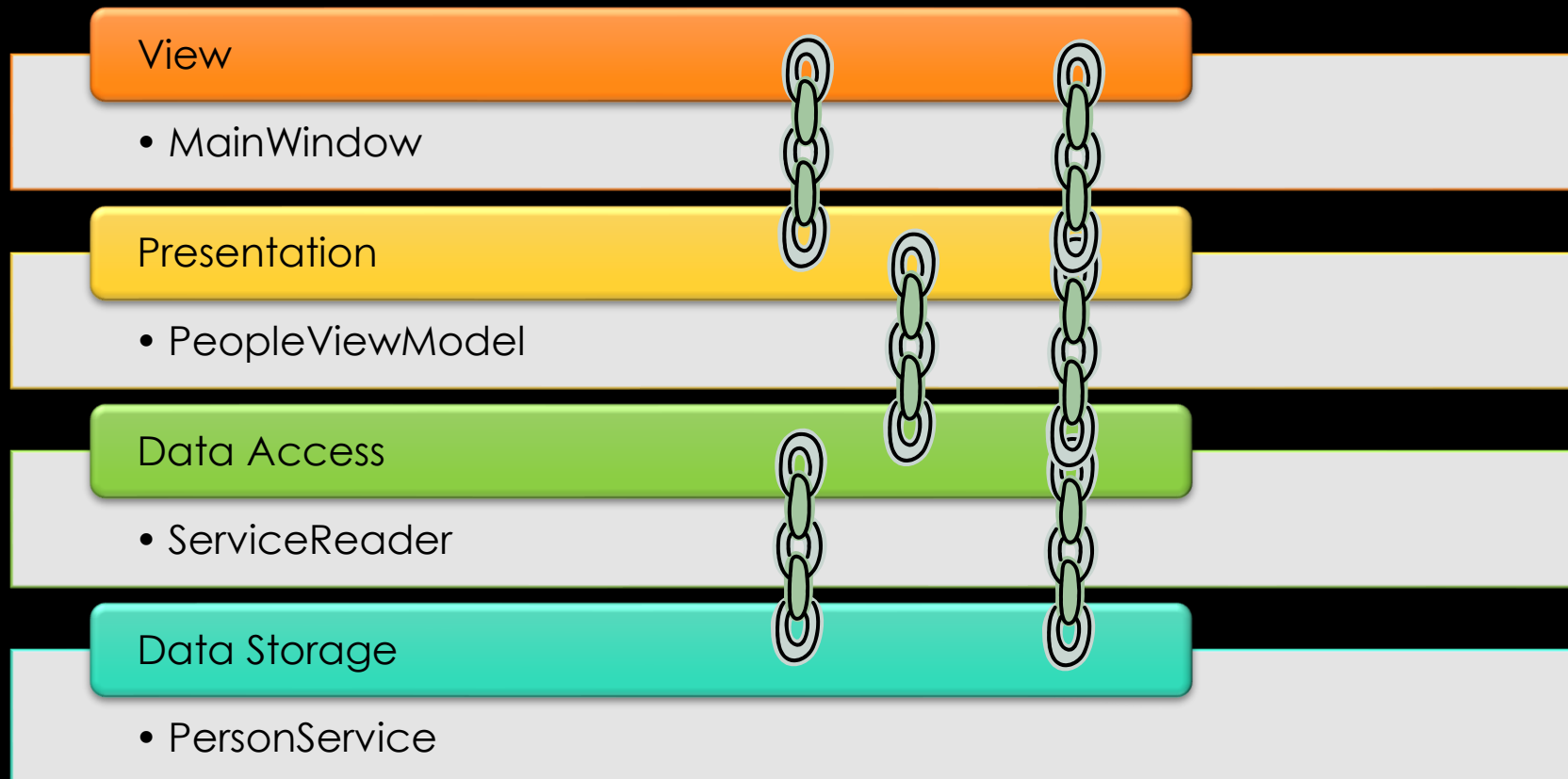
Tight Coupling



Creating a Caching Reader



Loose(r) Coupling



Dependency Injection Concepts

- DI Design Patterns
 - Constructor Injection*
 - Property Injection*
 - Method Injection
 - Ambient Context
 - Service Locator

- DI Containers
 - Unity
 - Castle Windsor
 - Ninject*
 - Autofac*
 - Spring.NET
 - and others

- Object Composition*

*Topics we'll touch on today

Primary Benefits

- Extensibility*
 - Late Binding*
 - Parallel Development
 - Maintainability
 - Testability*
-
- Adherence to S.O.L.I.D. Design Principles.

*Topics we'll touch on today



Thank You!

Jeremy Clark

- <http://www.jeremybytes.com>
- jeremy@jeremybytes.com
- @jeremybytes