

DI Why?

Getting a Grip on Dependency Injection

Jeremy Clark

www.jeremybytes.com

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

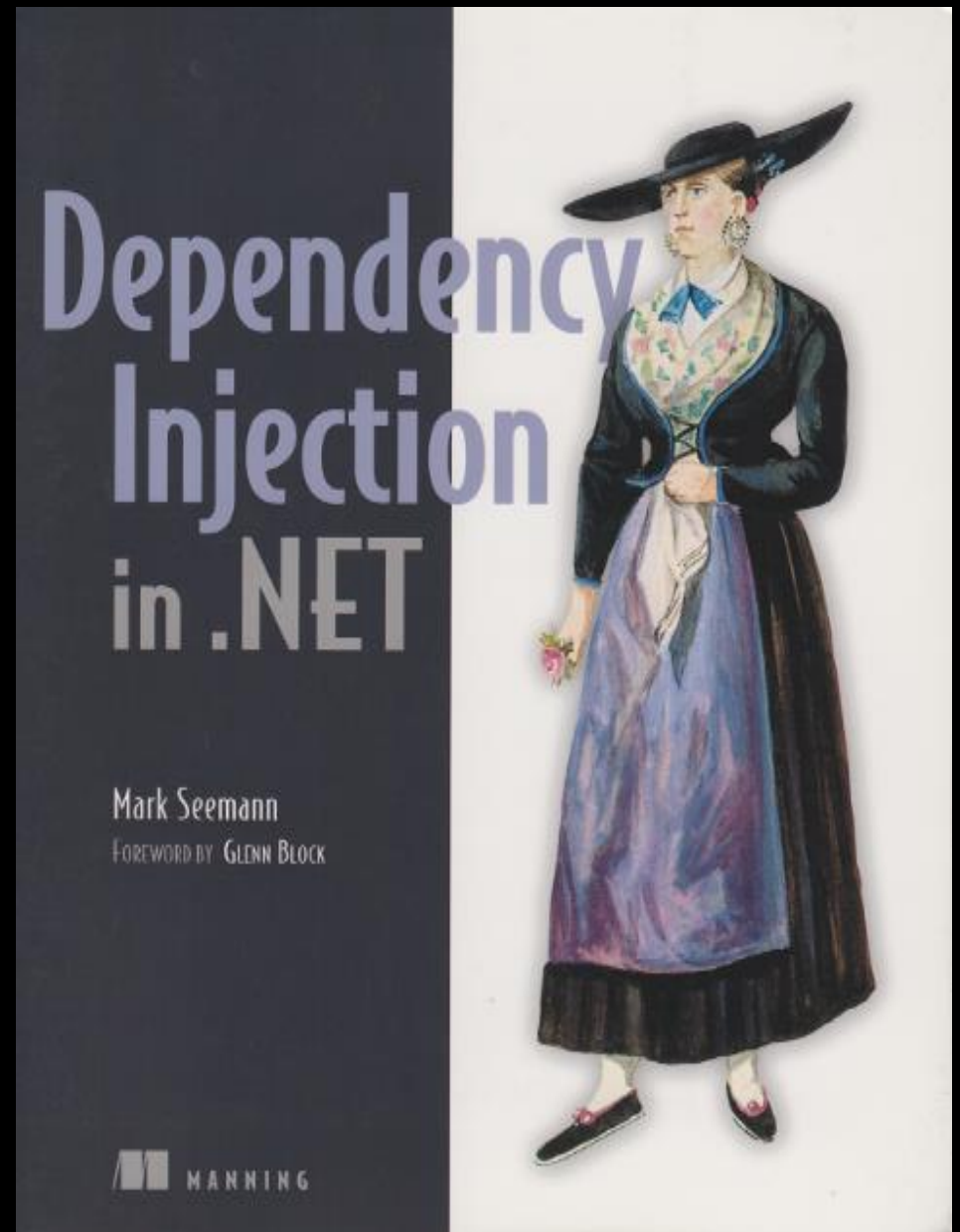
What Is Dependency Injection?

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

- Mark Seeman

Dependency Injection in .NET

- Mark Seeman



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

*Topics we'll touch on today

Application Layers

View

- MainWindow

View Model

- MainWindowViewModel

Repository

- PersonServiceRepository

Service

- PersonService

Look At The Code

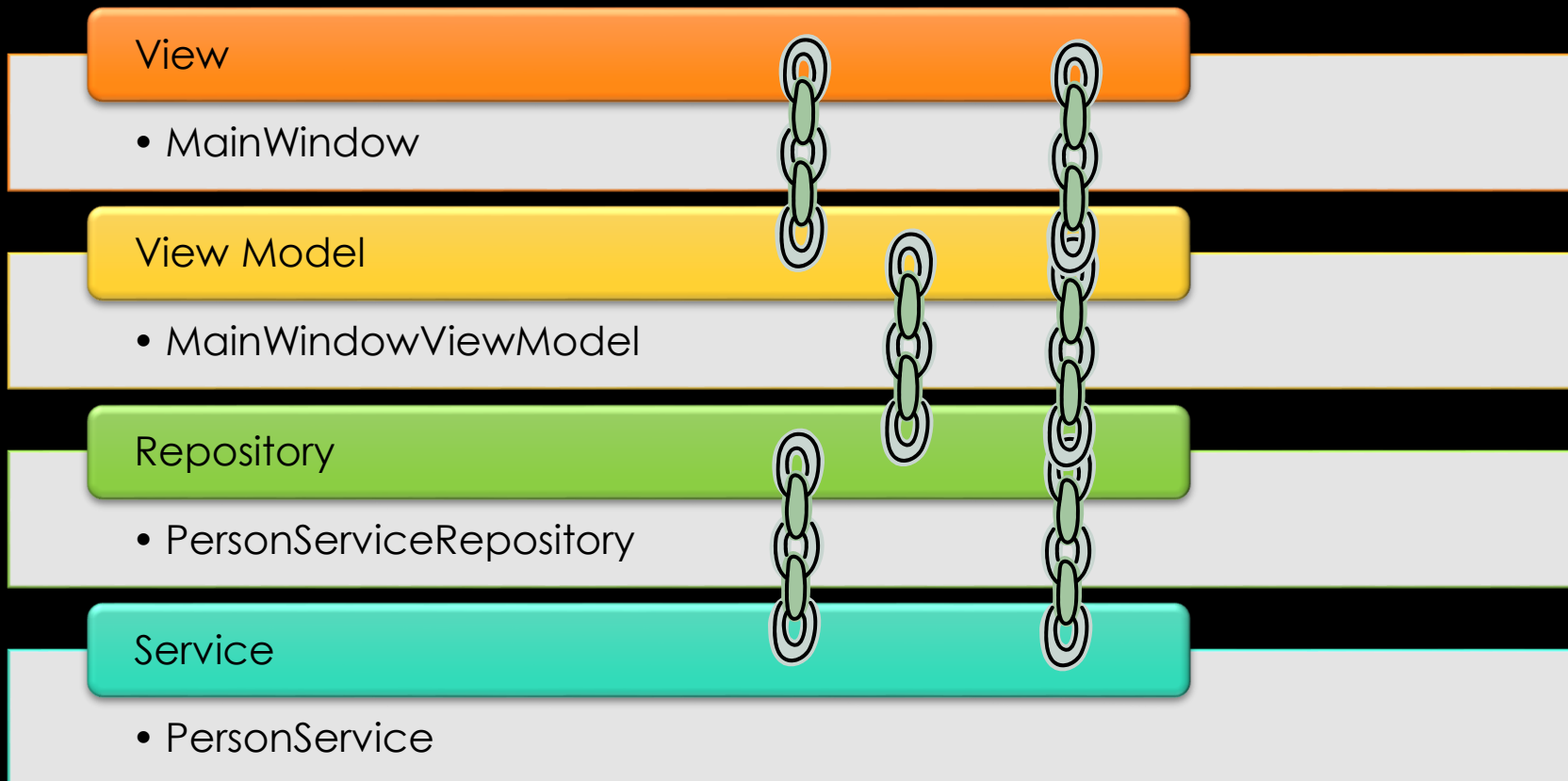
```
/// <summary>
/// Interaction logic for App.xaml
/// </summary>
public partial class App : Application
{
    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);
        ComposeObjects();
        Application.Current.MainWindow.Show();
    }

    private void ComposeObjects()
    {
        var wrappedRepository = new PersonServiceRepository();
        var repository = new CachingPersonRepository(wrappedRepository);
        var viewModel = new MainWindowViewModel(repository);
        Application.Current.MainWindow = new MainWindow(viewModel);
    }
}
```

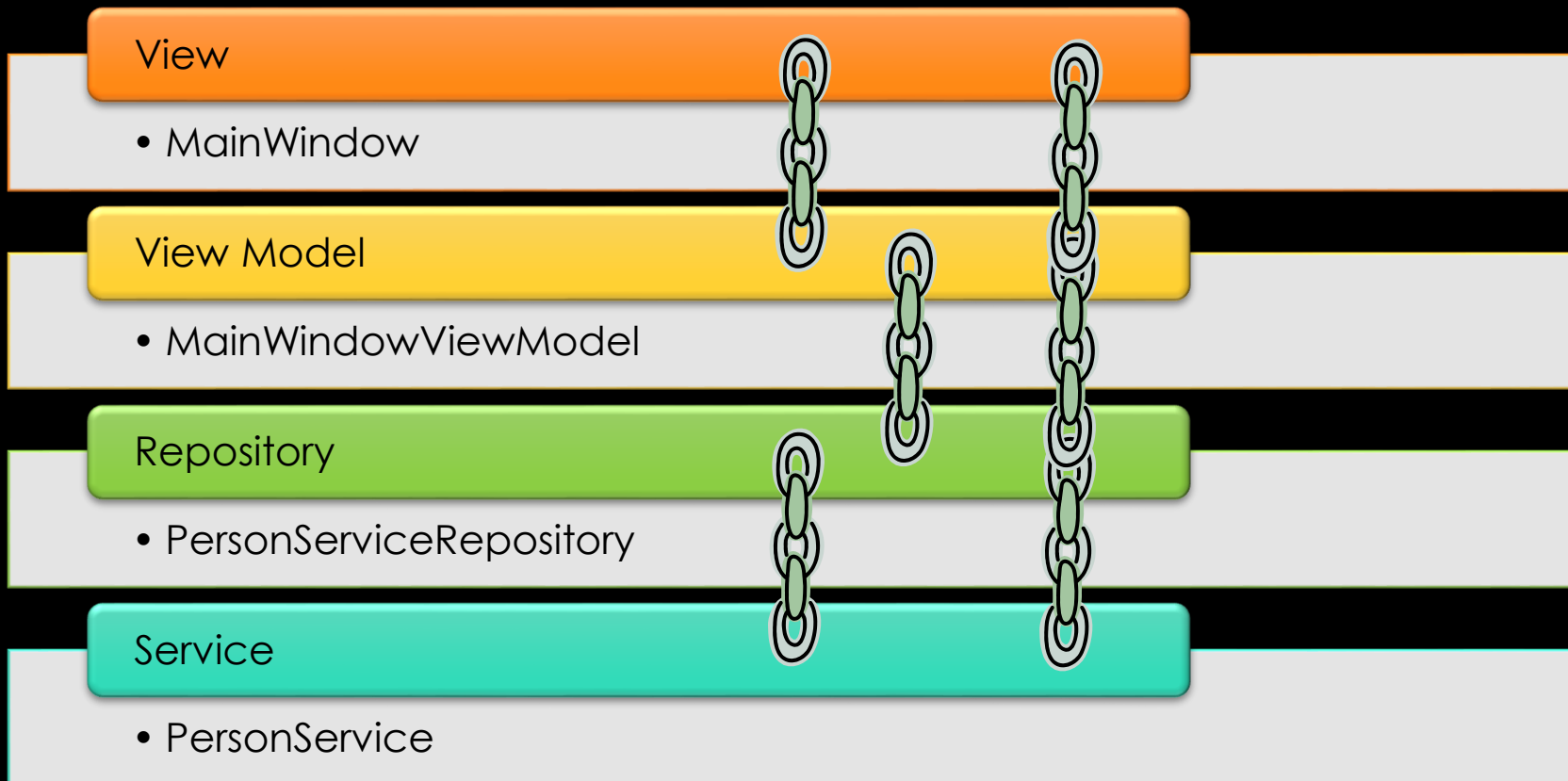
The screenshot shows the Visual Studio IDE with the following details:

- Window Title:** DependencyInjection - Microsoft Visual Studio
- Menu Bar:** File, Edit, View, Refactor, Project, Build, Debug, Team, Data, Tools, Architecture, Test, Analyze, Window, Help
- Toolbar:** Includes icons for File Explorer, Solution Explorer, and other development tools. The status bar shows "Windows Phone Emulator" and "Debug" mode.
- File Explorer:** Shows the project structure for "DependencyInjection" (11 projects). The selected project is "DI.UI.App", which contains files like App.xaml.cs, Converters.cs, and MainWindow.xaml.
- Code Editor:** Displays the C# code for App.xaml.cs, showing the OnStartup and ComposeObjects methods.
- Bottom Panel:** Includes tabs for Error List, Output, Find Results, Test Results, and Code Metrics Results. The status bar shows "Ready" and coordinates "Ln 1, Col 1, Ch 1, INS".

Tight Coupling



Loose(r) Coupling



Dependency Injection Concepts

- DI Design Patterns
 - Constructor Injection*
 - Property Injection*
 - Method Injection
 - Ambient Context
 - Service Locator
- Object Composition*
- DI Containers
 - Unity
 - Castle Windsor
 - Ninject*
 - Autofac
 - StructureMap
 - Spring.NET
 - and others

*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