What Is Dependency Injection?

And why does it look like it's making my application more complex?

David P. Donahue
Senior Consultant, Magenic Technologies
about.me/davidpdonahue

What Is Dependency Injection?

Why Would I Use It?

How Does It Simplify Code?

Show Me!

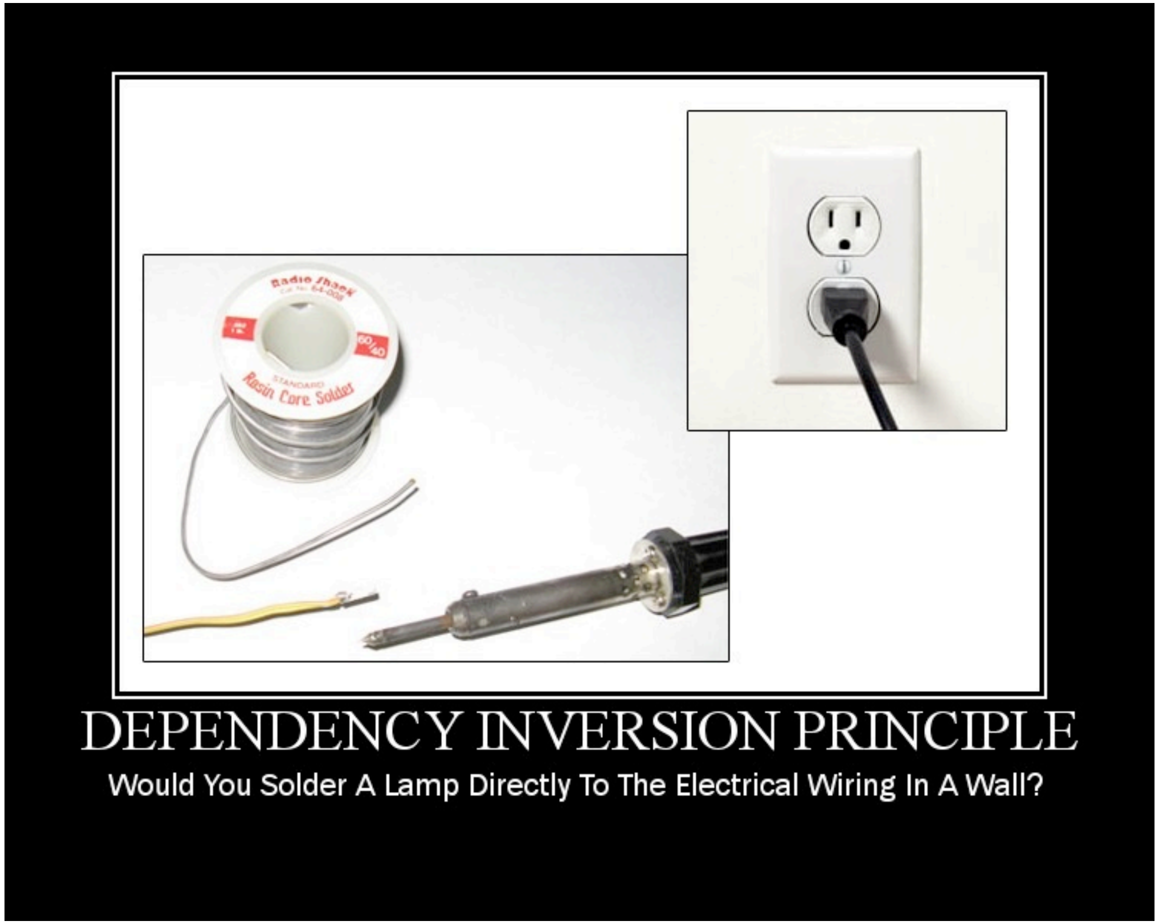
What Is Dependency Injection?

Why Would I Use It?

How Does It Simplify Code?

Show Me!

Dependency Inversion Principle



DEPENDENCY INVERSION PRINCIPLE

Would You Solder A Lamp Directly To The Electrical Wiring In A Wall?

http://lostechies.com/derickbailey/2009/02/11/solid-development-principles-in-motivational-pictures/

Abstractions should **not** depend upon details.

Details should depend upon abstractions.

http://en.wikipedia.org/wiki/Dependency_inversion_principle

“Require, Don’t Instantiate”

What Is Dependency Injection?

Why Would I Use It?

How Does It Simplify Code?

Show Me!

Unit Tests

Unit Tests

- Mock Data Access Layer

Unit Tests

- Mock Data Access Layer
- Mock File System

Unit Tests

- Mock Data Access Layer
- Mock File System
- Mock Email

Unit Tests

- Mock Data Access Layer
- Mock File System
- Mock Email
- etc.

Modular Design

Modular Design

Multiple Different Implementations

Modular Design

- Multiple Different Implementations
- Configurable Applications

Modular Design

- Multiple Different Implementations
- Configurable Applications
- Upgrade Infrastructure Components

Clean Separations

Clean Separations

Single Responsibility

Clean Separations

- Single Responsibility
- Increased Code Re-Use

Clean Separations

- Single Responsibility
- Increased Code Re-Use
- Cleaner Modeling

What Is Dependency Injection?

Why Would I Use It?

How Does It Simplify Code?

Show Me!

```
public class Computer {
  private PowerSupply _psu;

  public void PowerOn() {
      _psu = new PowerSupply(“400W”);
      _psu.PowerOn();
  }
}
```

```
public class Computer {
    private IPowerSupply _psu;

    public Computer(IPowerSupply psu) {
        _psu = psu;
    }

    public void PowerOn() {
        _psu.PowerOn();
    }
}
```

```
public class Computer {
    public IPowerSupply PSU { get; set; }

    public void PowerOn() {
        PSU.PowerOn();
    }
}
```

```
public class Computer {
    private IPowerSupply _psu
    public IPowerSupply PSU {
        get {
            if (_psu == null) _psu = IoC.Resolve<IPowerSupply>();
            return _psu;
        }
    }

    public void PowerOn() {
        PSU.PowerOn();
    }
}
```

What Is Dependency Injection?

Why Would I Use It?

How Does It Simplify Code?

Show Me!

Castle Windsor

http://www.castleproject.org/container/index.html

StructureMap

http://structuremap.net/structuremap/index.html

SPRING.net

Application Framework

http://www.springframework.net/

autofac

http://code.google.com/p/autofac/

Microsoft®

patterns & practices

proven practices for predictable results

http://unity.codeplex.com/

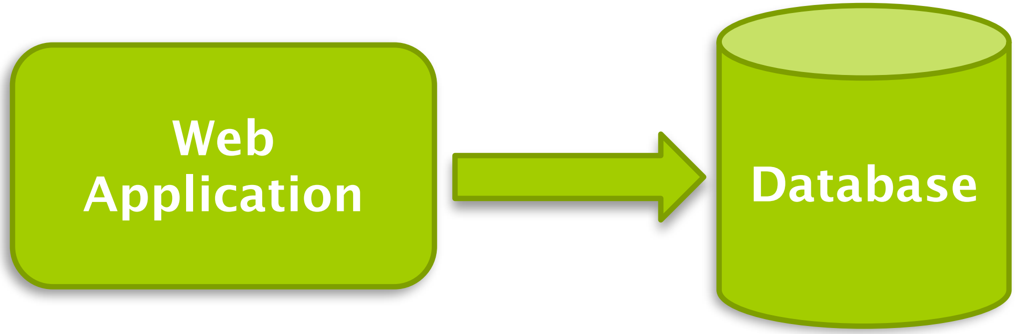
Ninject!

http://www.ninject.org/

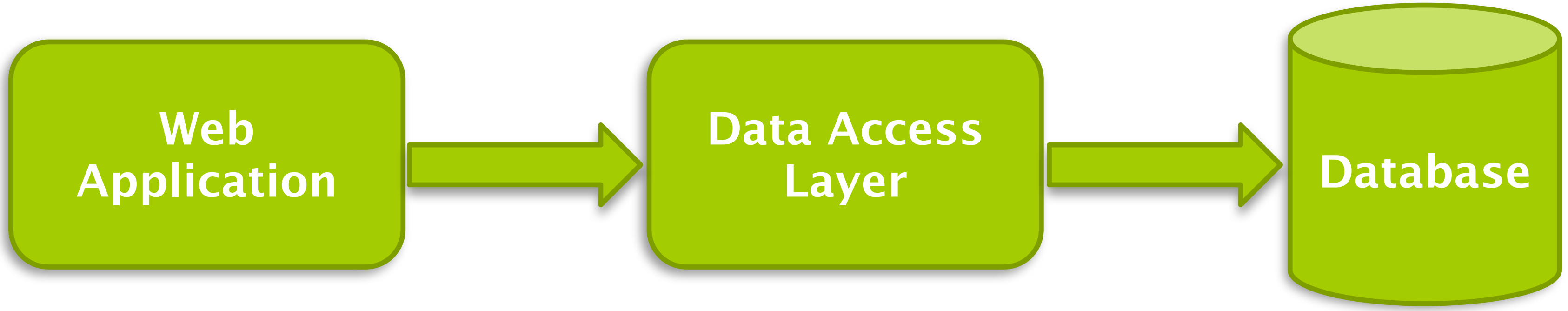
How About Some Actual Code?

Microsoft® Visual Studio®™

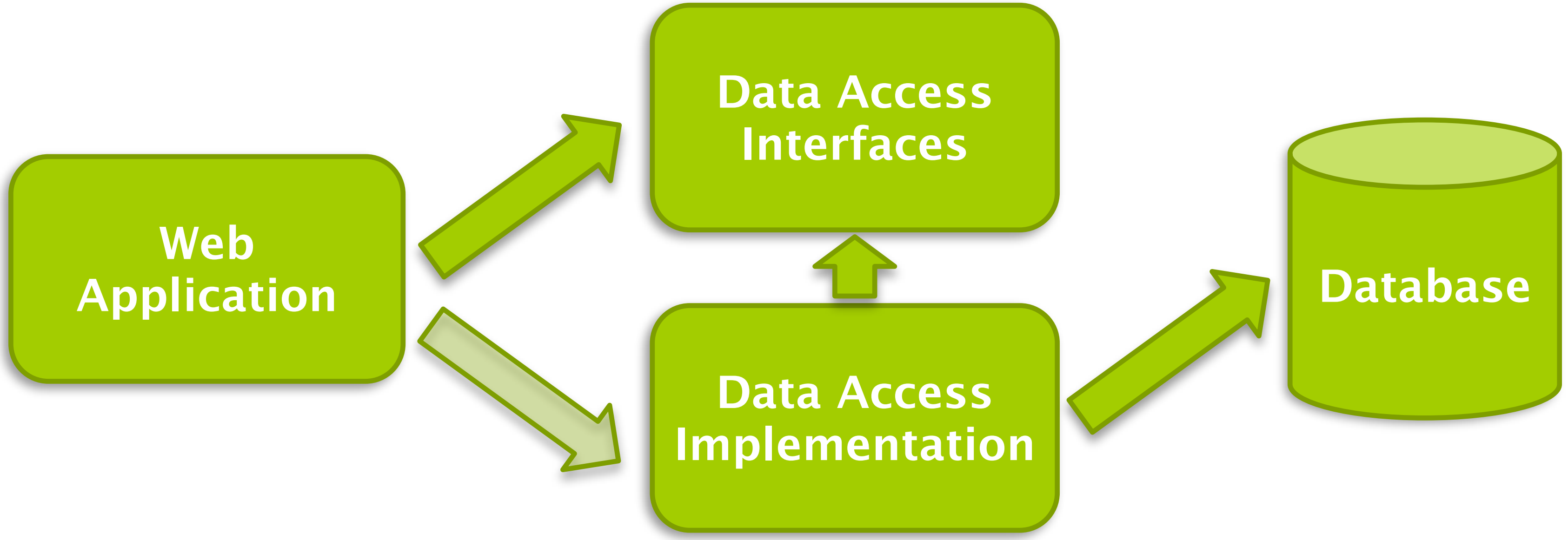
DISamples.NoLayers



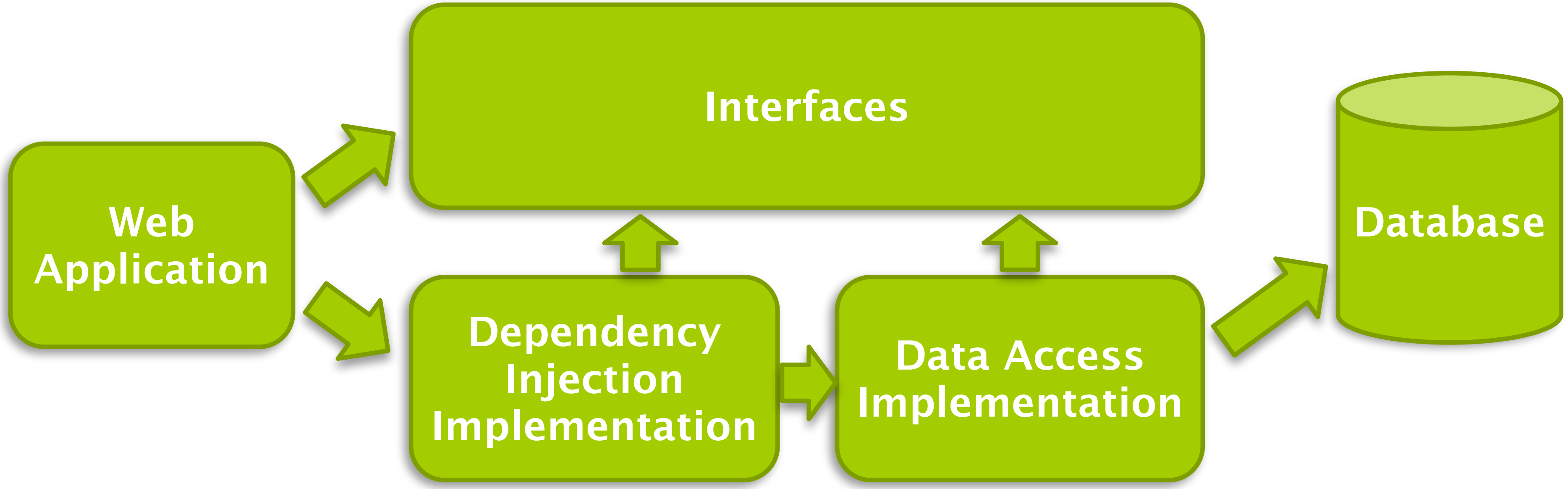
DISamples.Layers



DISamples.DI



DISamples.Domain


Interfaces
Web Application
Dependency Injection Implementation
Data Access Implementation
Database

Conclusion

Questions?

Comments?

Complaints?

David P. Donahue
Senior Consultant, Magenic Technologies
about.me/davidpdonahue