# Applied MVVM – Validation and Dependency Injection

**Brian Noyes**
CTO AND CO-FOUNDER, SOLLIANCE INC

@briannoyes www.briannoyes.com

# Introduction

Visual Studio data sources designer

Validation

Dependency injection

MVVM toolkits

# Visual Studio Data Sources Designer

Can use it to quickly scaffold data bound forms

Generates non-MVVM structured data binding hook-up

Can quickly morph into MVVM structure

# Validation in MVVM

**Data entry forms can still leverage WPF data binding validation features**

**Validation logic belongs in the Model or ViewModel, not the View**

**Can use any of:**

- Exceptions
- IDataErrorInfo
- INotifyDataErrorInfo
- ValidationRules

**Favor INotifyDataErrorInfo**

# Dependency Injection

Data binding decouples Views and ViewModels

Need something to decouple ViewModels from client services

Interfaces and dependency injection provide that decoupling

# Dependency Injection and Inversion of Control

**Inversion of Control (IoC) and Dependency Injection (DI) are closely related**

**A "container" is infrastructure code that does both for you**

**The container is responsible for:**

- Constructing an object when asked
- Determining what that object depends on
- Constructing those dependencies
- Injecting them into the object being constructed
- Recursively doing this process

**There are many DI libraries to choose from**

- Unity, AutoFac, Ninject, StructureMap, etc.

# MVVM Frameworks and Toolkits

Prism


MVVM Light


Caliburn Micro

MVVM Cross

# Prism Framework

MVVM

Modularity

UI composition/ regions

Navigation

Commands

Pub-sub events

# Summary

- Visual Studio Data Sources designer can scaffold out data-centric views quickly
- Validation logic belongs in the Model and/or ViewModel
- Dependency injection lets you keep ViewModels loosely coupled with Client Services
- Using a good MVVM Framework eliminates the need to write your own MVVM infrastructure code