Applied MVVM - Part 2

Brian Noyes

@briannoyes | www.solliance.net

Overview

Visual Studio Data 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

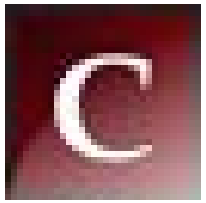
Dependency Injection / IoC Containers

- 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 Containers to choose from
  - Unity, AutoFac, Ninject, StructureMap, etc.

MVVM Toolkits / Frameworks

Prism

MVVM Light



Caliburn Micro

Prism

MVVM

Modularity

UI Composition / Regions

Navigation

Commands

Pub/Sub Events

Summary

- Visual Studio Data 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