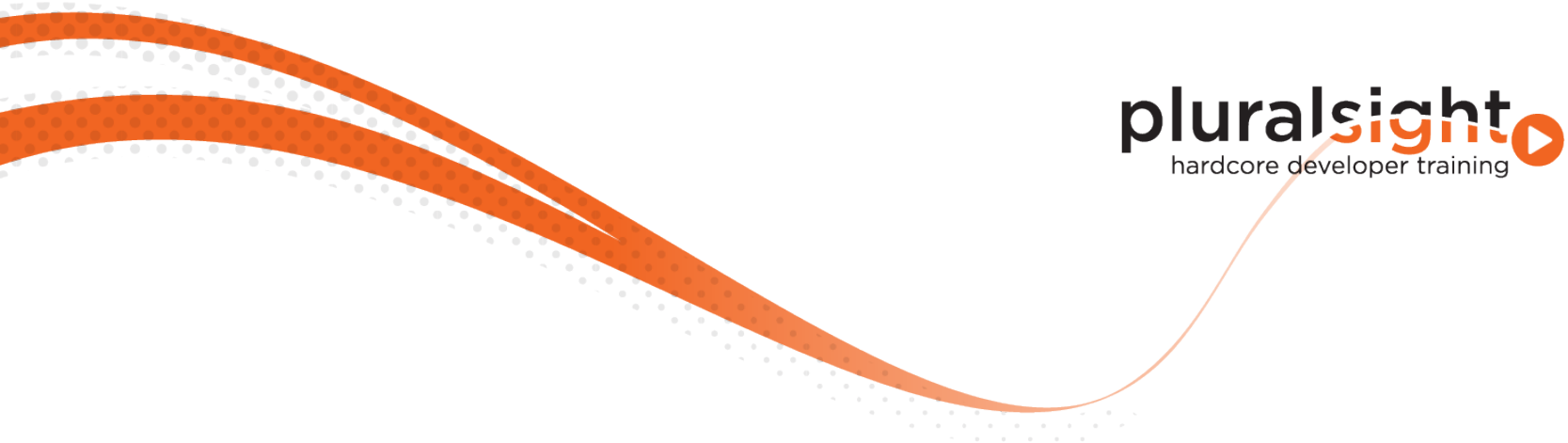
# Design-Time Data Patterns

Michael L Perry
Michael@qedcode.com



# Outline

- **Patterns**
  - In-Place Data
  - In-Place Child Items
  - Sample Data
  - Design-Mode View Models
  - Design Mode Models
  - Shadow Types

- **Goals**
  - Visualize application data
  - Stay in flow
  - Evolve with the application

# In-Place Data

## Stay in the designer

# Freedom to Brainstorm

# Realistic Data

## Brian Sullivan

Brian Sullivan is a senior consultant for Improving Enterprises in Dallas. He got his start in programming maintaining legacy mainframe applications in COBOL at a large trucking company, but quickly realized he needed to find a more productive environment in order to stay sane. He jumped at the opportunity to help transition some of those COBOL applications to .NET, and he hasn’t looked back since. He has

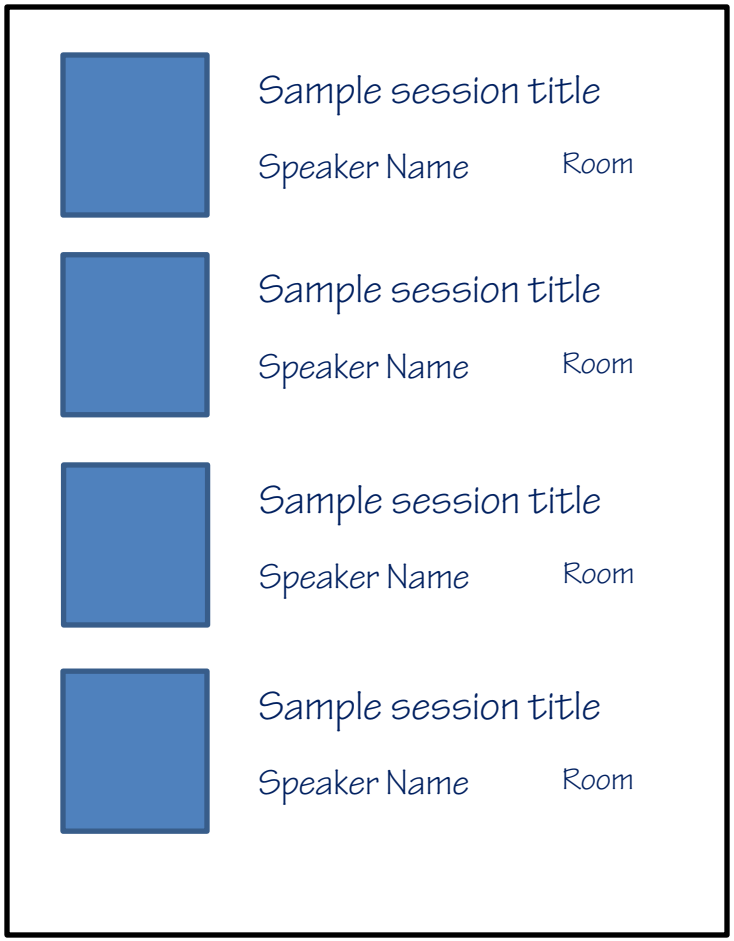
# Paste into XAML

- **Evaluate**
  - Weight
  - Emphasis
  - Relationships
  - Size
  - Color
  - Proximity

# In-Place Child Items

## Vary list items

# Items Controls



# In-Place Strings

```
xmlns:s="clr-namespace:System;assembly=mscorlib"

<DataTemplate x:Key="SessionItemTemplate">
    <TextBlock Text="{Binding}" />
</DataTemplate>

<ListBox ItemTemplate="{DynamicResource SessionItemTemplate}">
    <s:String>ASP.NET - Don't Do THAT! </s:String>
    <s:String>Building an End-to-End HTML5 App</s:String>
    <s:String>Data Binding Improvements 4.5</s:String>
</ListBox>
```

# Complex Types

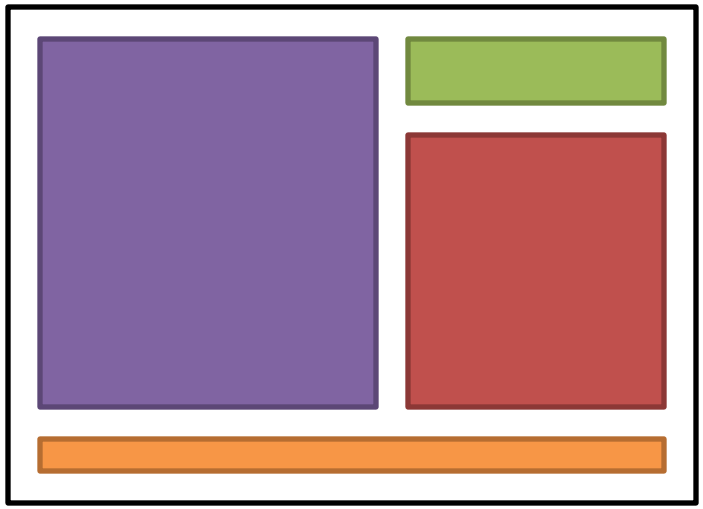
- Define data types
- Auto properties
- Set attributes
- Bind to attributes

- Not a real view model!

# Sample Data

## Remove data from XAML

# Add Structure



MVVM

# Sample Data in Blend

- Stored in a separate XML file
- Bound to views at design time
- Does not work in Windows Store apps (yet)

# Generate Data Per Property

- **Lorem Ipsum by default**
- **Change type:**
  - Name
  - Phone Number
  - Address
  - Email
  - …
- **Not representative of domain**

# Based on View Model

- **View Models change**
- **Regenerate sample data**
  - Change the rules
    - Word count
    - Word length
  - Delete and recreate
    - Re-bind to container
    - Child bindings are preserved

# Do Not Edit!

- Easy to regenerate
- Takes you out of flow
- Better patterns for realistic data

# Design-Mode View Models

## Realistic data

# Created in Code

- **Designer executes the code**
- **Stateful view models**
- **Steps**
  - Design-mode data source
  - Resource dictionary
  - d:DataContext
- **View model locator for the designer**

# Tradeoff

- **More realistic**
  - Not lorum ipsum
- **More expensive**
  - Wait until view models stabilize

# Design-Mode Models

## Stateless view models

# Stateless View Models

- **Benefits**
  - Fewer degrees of freedom
  - Data can’t get out-of-sync
  - Less plumbing
- **Cost**
  - Design-mode data is harder
  - No state: reference to model
  - Create a design-mode model

# Design-Mode Data Source

- Create models in constructor
- Store models in fields
- Inject models into stateless view models

# Tradeoff

- **Reuse models**
  - Consistency in designer
- **Exposes view model behavior**
  - Data formatting
- **Harder to maintain**

# Shadow Types

Avoid view model dependencies

# No Type Checking

- **No type declaration on DataContext**
- **Walks the Path**
- **Same shape as view model**
  - Shadow types

# Anonymous Shadow Types

```
public class DesignModeDataSource
{
    public object Property
    {
        get
        {
            return new
            {
                PropertyA = "Value A",
                PropertyB = "Value B",
                PropertyC = "Value C"
            };
        }
    }
}
```

# Explicit Shadow Types

```
public class ShadowType
{
    public string PropertyA { get; set; }
    public string PropertyB { get; set; }
    public string PropertyC { get; set; }
}
```

```
public class DesignModeDataSource
{
    public object Property
    {
        get
        {
            return new ShadowType
            {
                PropertyA = "Value A",
                PropertyB = "Value B",
                PropertyC = "Value C"
            };
        }
    }
}
```

# Tradeoff

- Realistic data
- No view model dependencies

- No model reuse
- No view model behavior

# Summary

- **Goals**
  - Visualize application data
  - Stay in flow
  - Evolve with the application

- **Patterns**
  - In-Place Data
  - In-Place Child Items
  - Sample Data
  - Design-Mode View Models
  - Design Mode Models
  - Shadow Types