Patterns for Maintainable Code

Mark Heath
http://markheath.net
@mark_heath

pluralsight
hardcore dev and IT training

Maintainable Code

Are Windows Forms applications difficult to maintain?

Monolithic MainForm

Many Controls

Everything in "Code Behind"

The Maintainability Problem

The problem with monolithic MainForm.cs …

Hard to **comprehend**

Hard to **test**

Hard to **reuse**

The Maintainability Solution

Segregate your user interface

Extract business logic from code behind

Create passive **Views** controlled by **Presenters**

Use the **Command Pattern** for buttons

Use an **Event Aggregator** for messaging

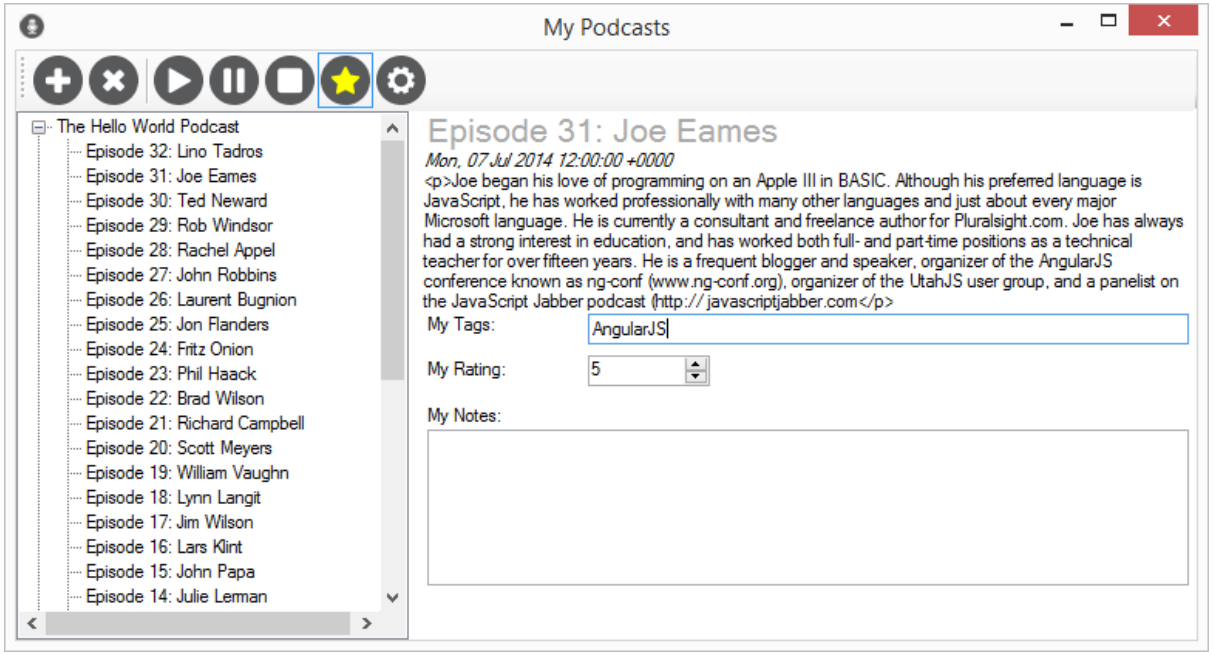
Segregating Your User Interface

```
private System.Windows.Forms.Panel panel1;
public System.Windows.Forms.NumericUpDown numericUpDownRating;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label1;
public System.Windows.Forms.TextBox textBoxNotes;
public System.Windows.Forms.TextBox textBoxTags;
private System.Windows.Forms.FlowLayoutPanel flowLayoutPanel1;
public System.Windows.Forms.Label labelEpisodeTitle;
public System.Windows.Forms.Label labelPublicationDate;
public System.Windows.Forms.Label labelDescription;
private System.Windows.Forms.ToolTip toolTip1;
```

Segregate your user controls from the outset

Segregating Your User Interface

Toolbar View



Subscriptions View

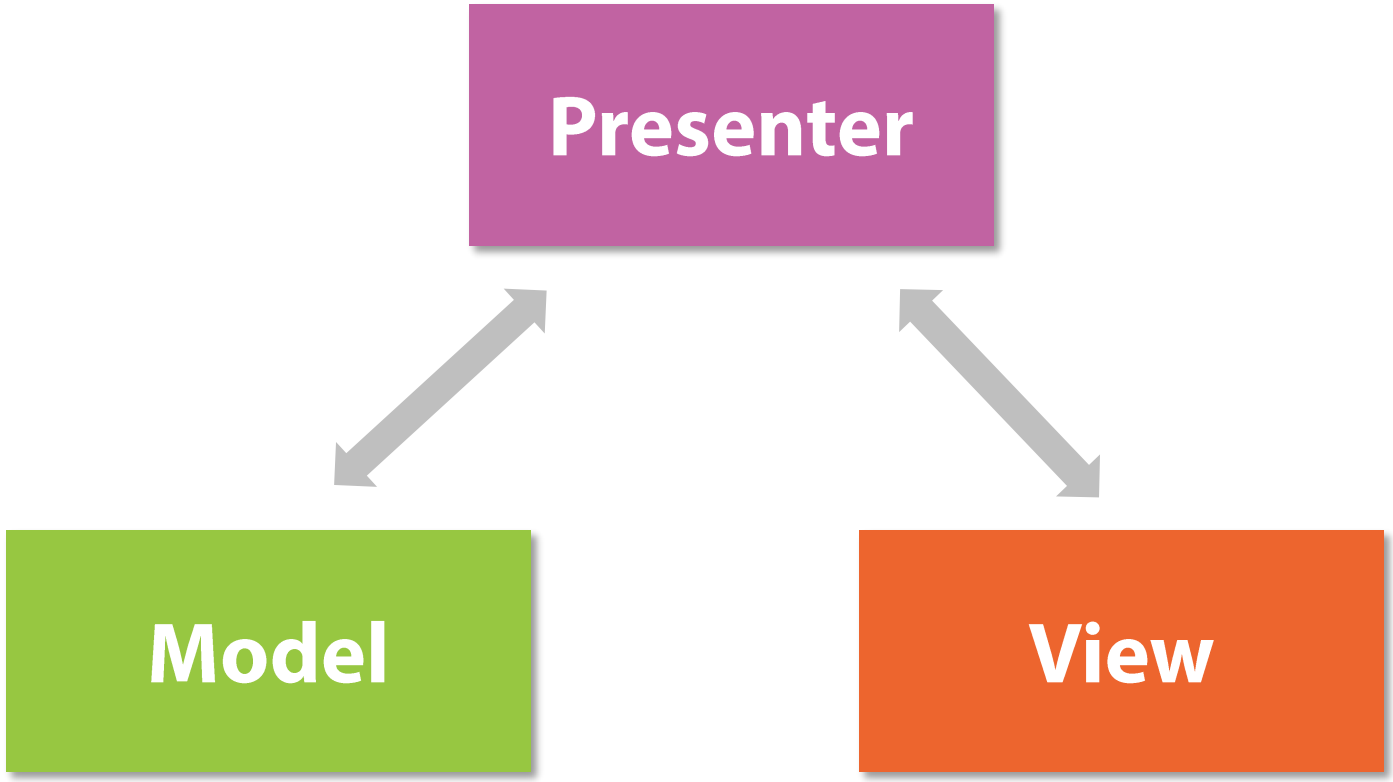
Episode View

(or Podcast View)

Extract Business Logic from Code Behind

- Identify code that doesn't belong in the code behind
- Extract it into business logic classes with a single responsibility

Model View Presenter



The Command Pattern

Create a class to represent each command in your application

PlayCommand

Execute

IsEnabled

IsEnabledChanged

IsVisible

Icon

ToolTip

ShortcutKey

CommandBase

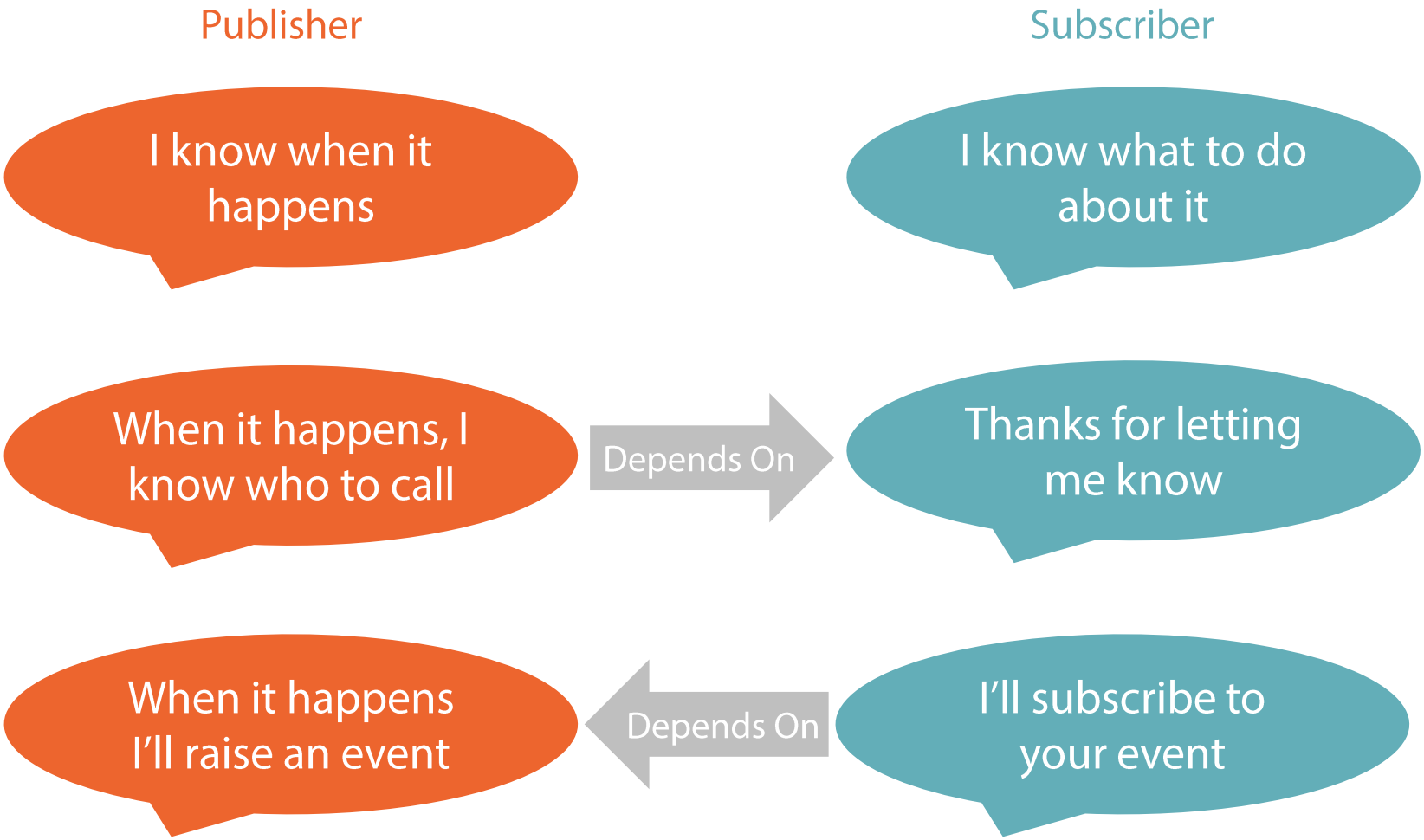
Exception Handling

Privileges

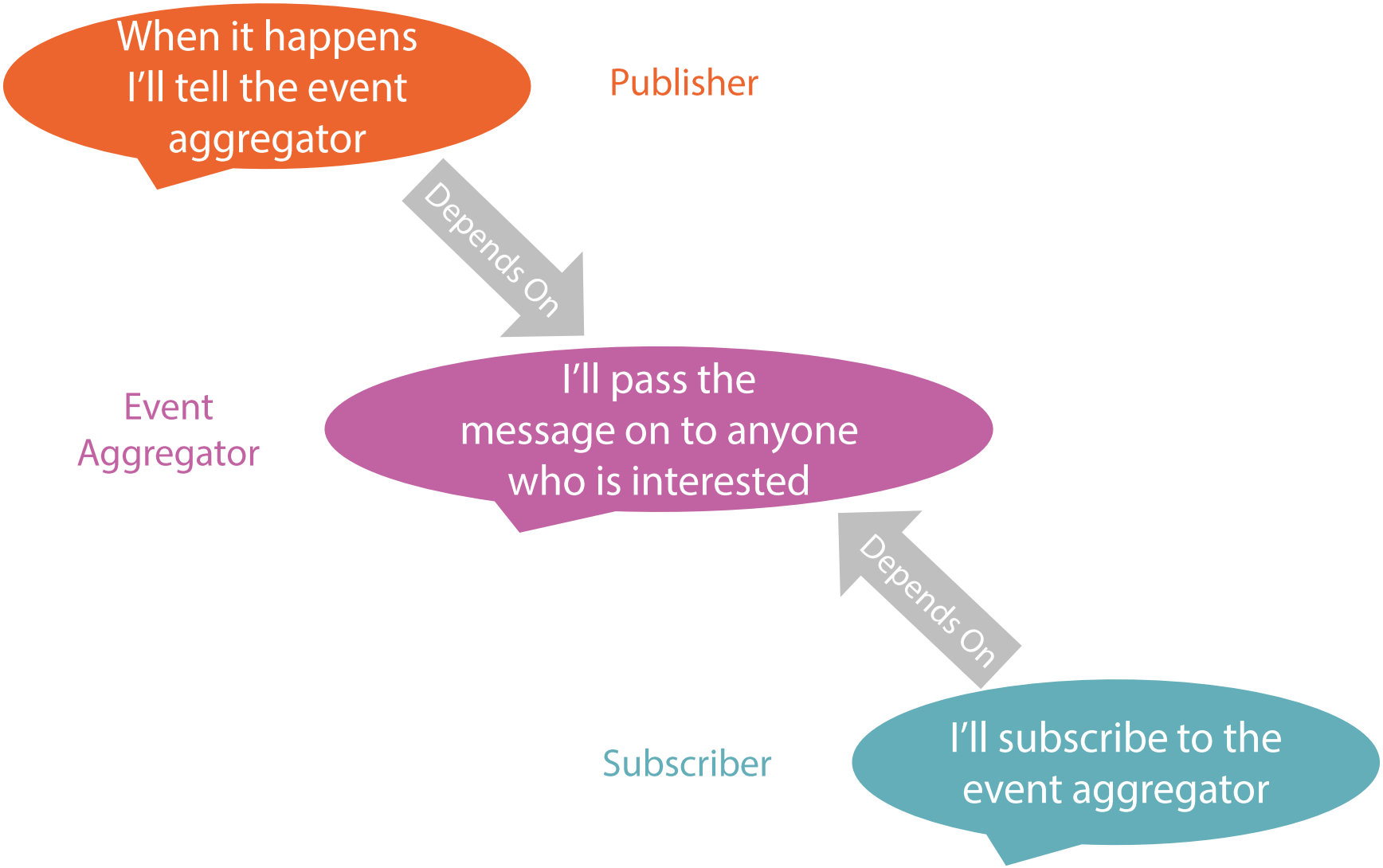
Licensing

Enabled Checking

Event Publishers and Subscribers



Event Aggregator



Module Summary

Segregate your user interface

Extract business logic from code behind

Create passive **Views** controlled by **Presenters**

Use the **Command Pattern** for buttons

Use an **Event Aggregator** for messaging

Maintainable Windows Forms Code

Introduce these patterns as early as possible

Easy to **comprehend**

Easy to **test**

Easy to **reuse**

Learning More

- **Patterns Library**
  - http://pluralsight.com/courses/patterns-library
  - Model View Presenter
  - Command Pattern
  - Event Aggregator
- **Inversion of Control Containers**
  - http://pluralsight.com/courses/inversion-of-control