

# Moving Beyond the Basics with Xamarin.Forms

---

EXTENDING CONTROLS TO DO MORE THAN  
ACCEPT DATA



**Matthew Soucoup**

PRINCIPAL

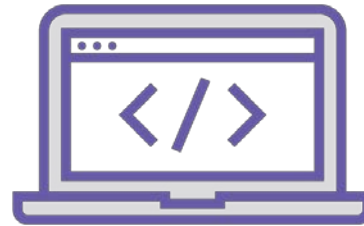
@codemillmatt codemilltech.com



# Xamarin.Forms Is



**Shared  
user interface**



**Shared  
application logic**



**Platform  
specific features**



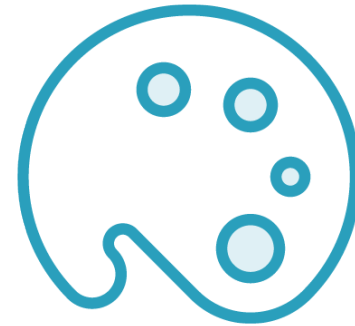
# Xamarin.Forms Apps Need To



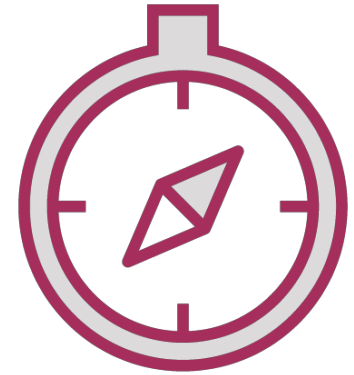
**Accept and  
transform input**



**Display varying,  
dynamic data**



**Maintain a  
consistent look**



**Embed native  
UI elements**

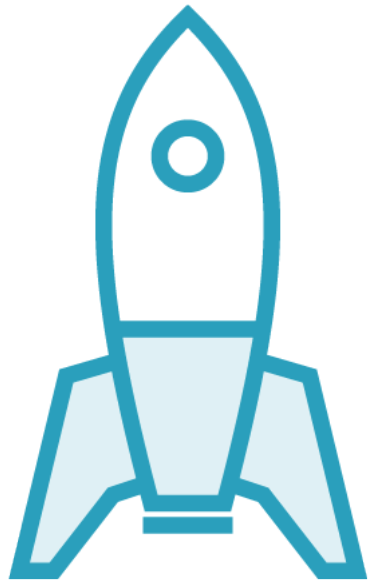




Xamarin.Forms Apps Need  
Reusable Components



# Xamarin.Forms Beyond the Basics



**Previous Xamarin.Forms experience**

**Examples in XAML**

**iOS and Android**

**Demo app from scratch**



# User Input Topics



## Setting up a TableView

- Built-in cells

## Creating custom cells

## Extending functionality with Behaviors

## Reacting to changes with Triggers



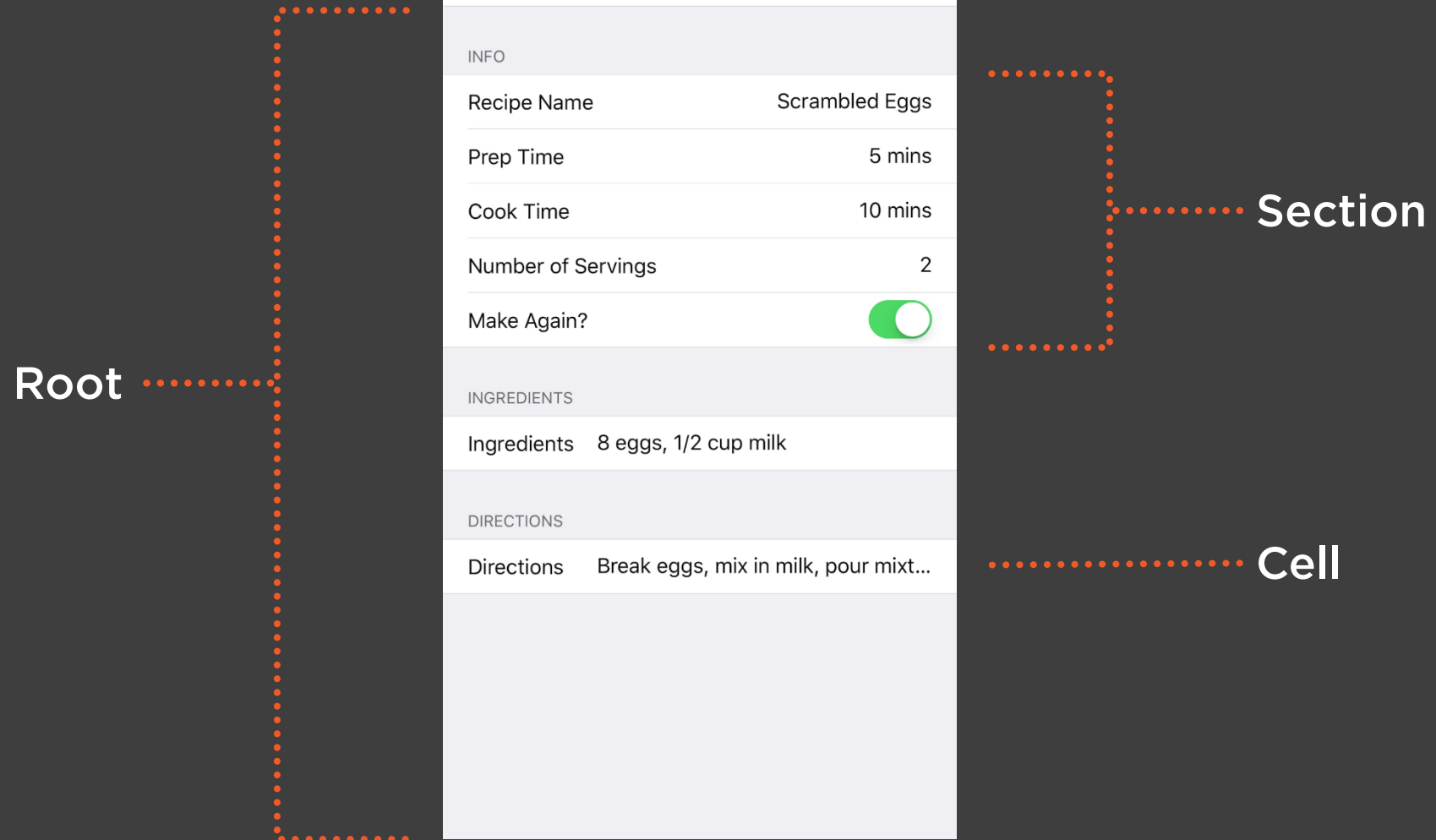
# The TableView & Working with Built-in Cells

---



# TableView

Root



Foodie	
INFO	
Recipe Name	Scrambled Eggs
Prep Time	5 mins
Cook Time	10 mins
Number of Servings	2
Make Again?	<input checked="" type="checkbox"/>
INGREDIENTS	
Ingredients	8 eggs, 1/2 cup milk
DIRECTIONS	
Directions	Break eggs, mix in milk, pour mixt...

Section

Cell





# TableView

Root

Foodie

Info

Recipe Name Scrambled Eggs

Prep Time 5 mins

Cook Time 10 mins

Number of Servings 2

Make Again? ☒

Ingredients

Ingredients 8 eggs, 1/2 cup milk

Directions

Directions Break eggs, mix in milk, pour mixture into

Section

Cell



# TableView Intent



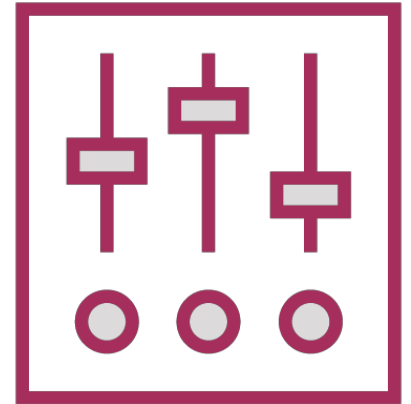
Data



Form

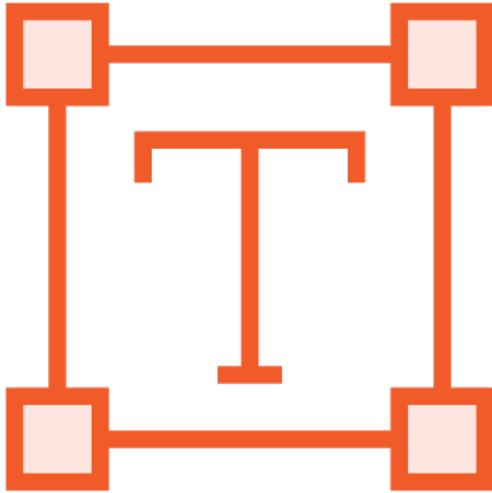


Settings



Menu

# Built-in Editing Cells



Entry cell



Switch cell

# Demo



**Setting up a TableView**

**Adding a Section**

**Adding built-in cells**



# TableView Summary



Accepts data

Specific Intent

Entry cell

Switch cell

# Custom TableView Cells

---



# Custom Cells

Make again?

Ingredients

Ingredients

8 eggs, 1/2 cup milk

Directions

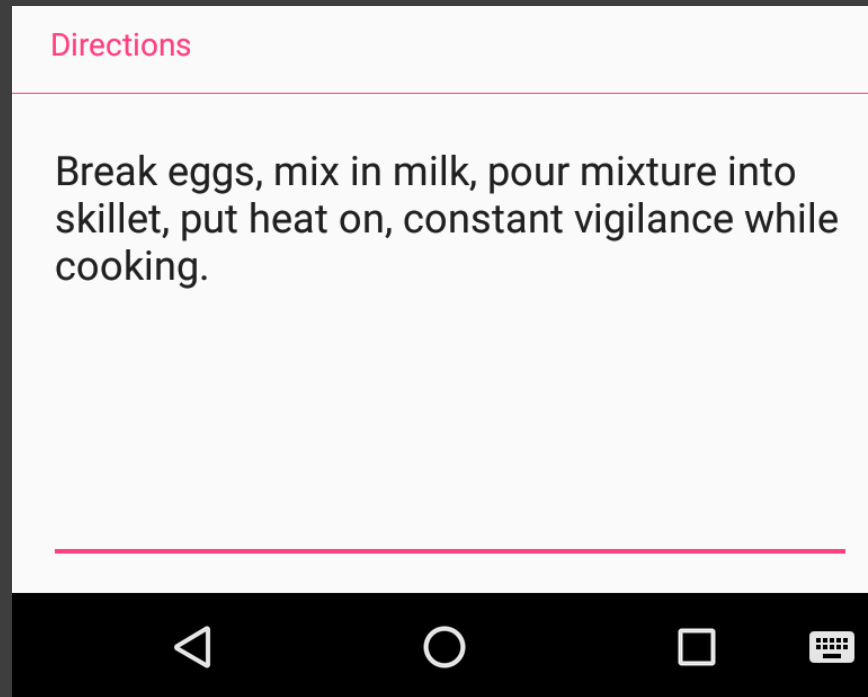
Directions

Break eggs, mix in milk, pour mixture

Not enough room



# Custom Cells



Enough room

ViewCell





```
<TableView HasUnevenRows="true">
```

```
...
```

```
<TableViewCell>
```

```
  <StackLayout
```

```
    Orientation="Horizontal">
```

```
    <Label
```

```
      Text="Recipe Name" />
```

```
    <Entry
```

```
      Text="Scrambled Eggs" />
```

```
  </StackLayout>
```

```
</TableViewCell>
```

◀ Multiple row heights

◀ Build custom visual layout



# Demo



Adding custom cells

Handling different types of layouts



# Custom Cells Summary



**Advanced use cases**

**Build within a ViewCell**

**Any visual layout**

**Remember HasUnevenRows**



# Behaviors

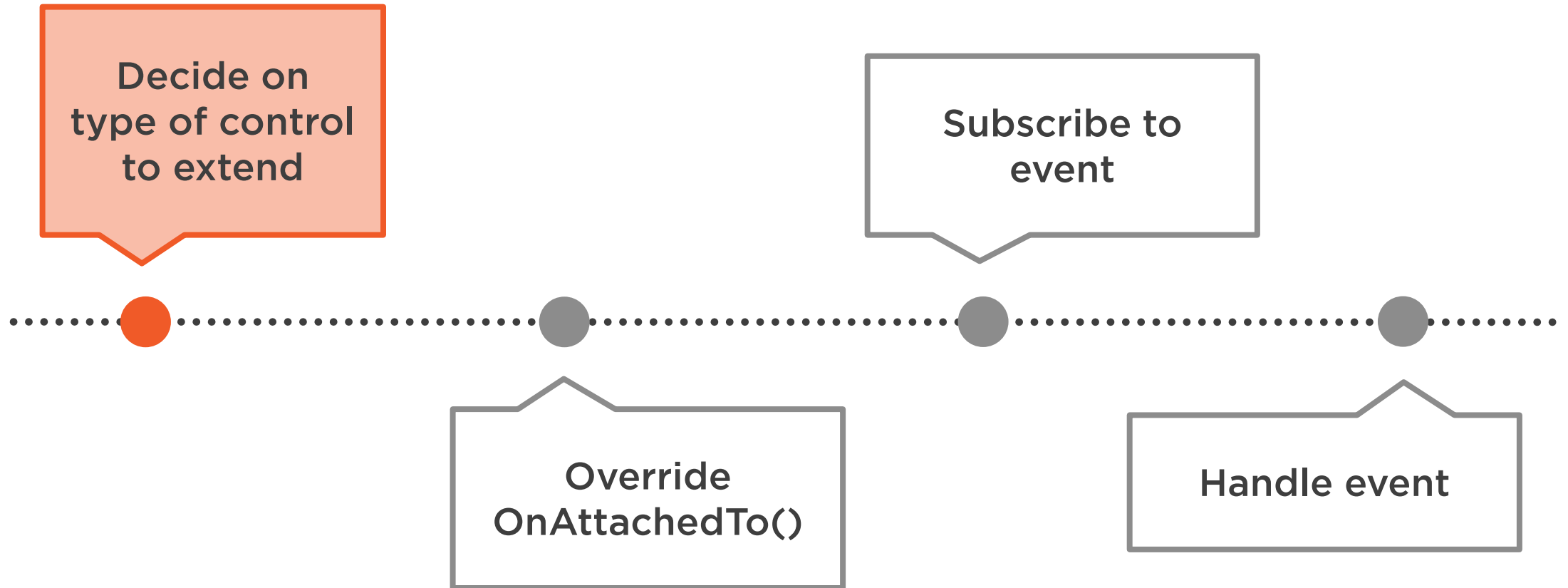
---



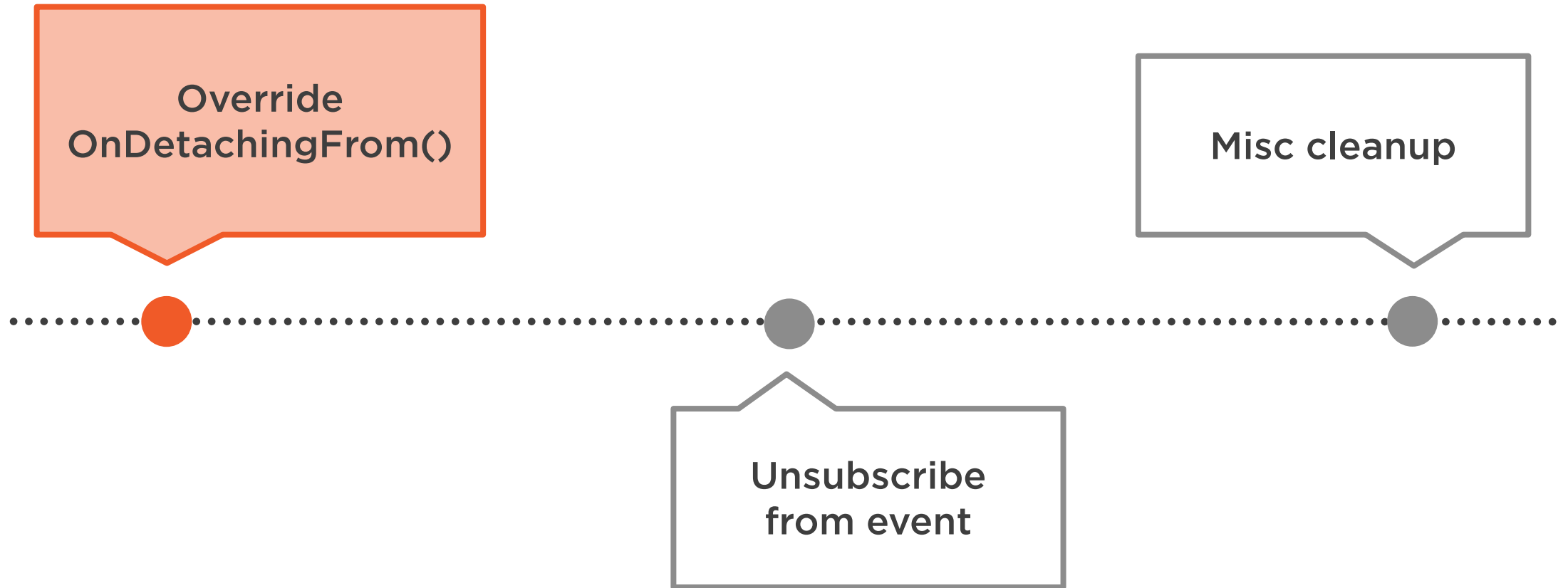
# Behavior



# Implementing a Behavior - Setup



# Implementing a Behavior – Tear Down



# Demo



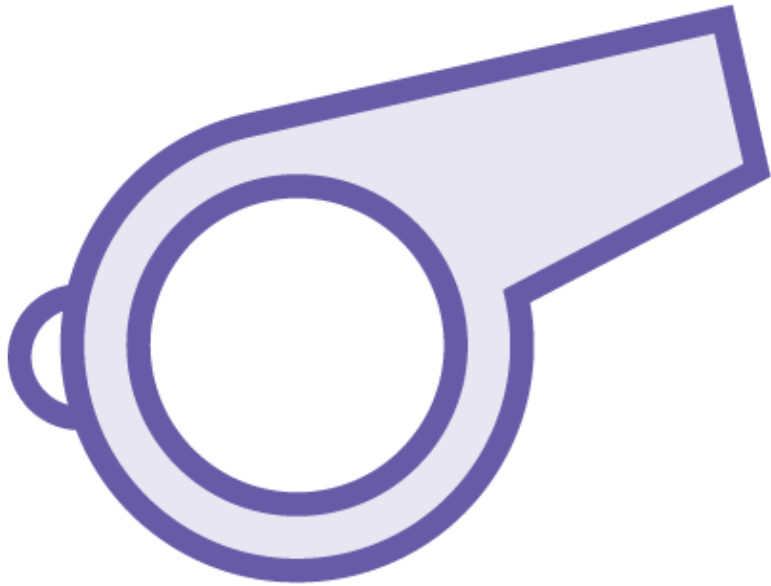
Creating a Behavior

Consuming a Behavior





# Behavior Summary



**Extend control functionality**

**Can add multiple to a control**

**Reusable**

**Handle the events to extend**

**Cleanup after events**

# Triggers

---



# Triggers



# Trigger Types

## Property

Respond to  
property state

## Data

Respond to  
another control

## Event

Respond to an  
event firing



```
<Editor.Triggers>
  <Trigger TargetType="Editor" Property="IsFocused" Value="True">
    <Setter Property="BackgroundColor" Value="#FFF9C4" />
  </Trigger>
  <Trigger TargetType="Editor" Property="IsFocused" Value="True">
    <Setter Property="FontAttributes" Value="Bold" />
  </Trigger>
</Editor.Triggers>
```

# Property Triggers

## Trigger

- TargetType, Property, Value

## Setter

- Property, Value



```
<Editor.Triggers>
  <Trigger TargetType="Editor" Property="IsFocused" Value="True">
    <Setter Property="BackgroundColor" Value="#FFF9C4" />
    <Setter Property="FontAttributes" Value="Bold" />
  </Trigger>
</Editor.Triggers>
```

# Property Triggers

**Multiple setters per trigger**



```
<Label.Triggers>
    <DataTrigger TargetType="Label"
        Binding="{Binding Source={x:Reference recipeNameEntry},
            Path=IsFocused}" Value="True">
        <Setter Property="TextColor" Value="#D32F2F" />
    </DataTrigger>
</Label.Triggers>
```

## Data Triggers

**Bind to property on other control**

**DataTrigger class name**



```
public class RequiredValidationTriggerAction : TriggerAction<Entry>
{
    protected override void Invoke(Entry sender)
    {
        sender.BackgroundColor =
            string.IsNullOrEmpty(sender.Text) ?
            Color.FromHex("#FFCDD2") : Color.Default;
    }
}
```

## Event Triggers - Creation

**Implement in code - subclass TriggerAction**

**Override Invoke()**

- Check property or take action





```
xmlns:local="clr-namespace:Foodie"
```

```
...
```

```
<Entry.Triggers>  
  <EventTrigger Event="TextChanged">  
    <local:RequiredValidationTriggerAction />  
  </EventTrigger>  
</Entry.Triggers>
```

## Event Triggers - Consume

**Define namespace**

**Define event to react to**

**No need for TargetType**



# Demo



## Create the three types of triggers

- Property
- Data
- Event



# Triggers Summary



## **React to changes**

- Within same control – property
- Different control – data
- Respond to events – event

**Many per control possible**

**Many setters per trigger**



Table views organized way to accept data

Custom cells extend built-in functionality

Behaviors extend control functionality

Triggers react to changes

