Getting hands-on with XAML and Xamarin.Forms

Mitch Muenster
@MobileRez

What is this about?

- Using Xamarin.Forms
- Working with XAML Syntax & Behavior
- Using Advanced XAML
- Doing more then just a Tech talk but letting you code

Using Xamarin.Forms

What is Xamarin.Forms?

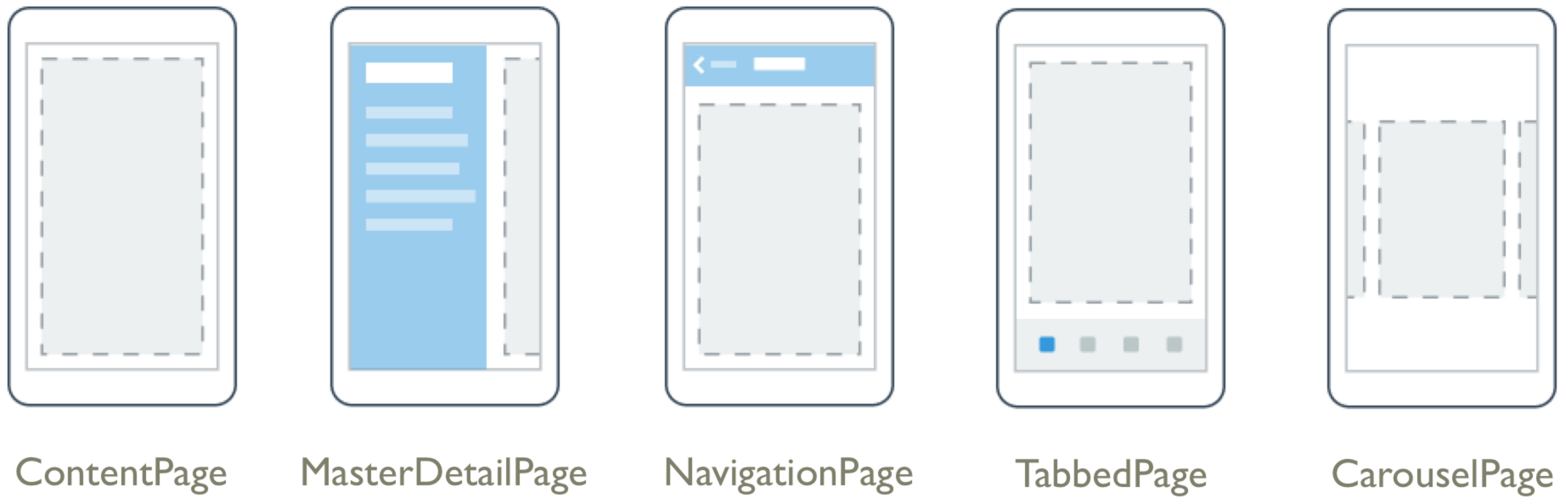
- Xamarin.Forms is a software framework that lets you build one app with UI and behavior and deploy it to each of the mobile platforms

Understanding Xamarin.Forms UI

- Xamarin.Forms UI is defined in 4 different ways; Pages, Layouts, Cells, and Views.

What is a Page?

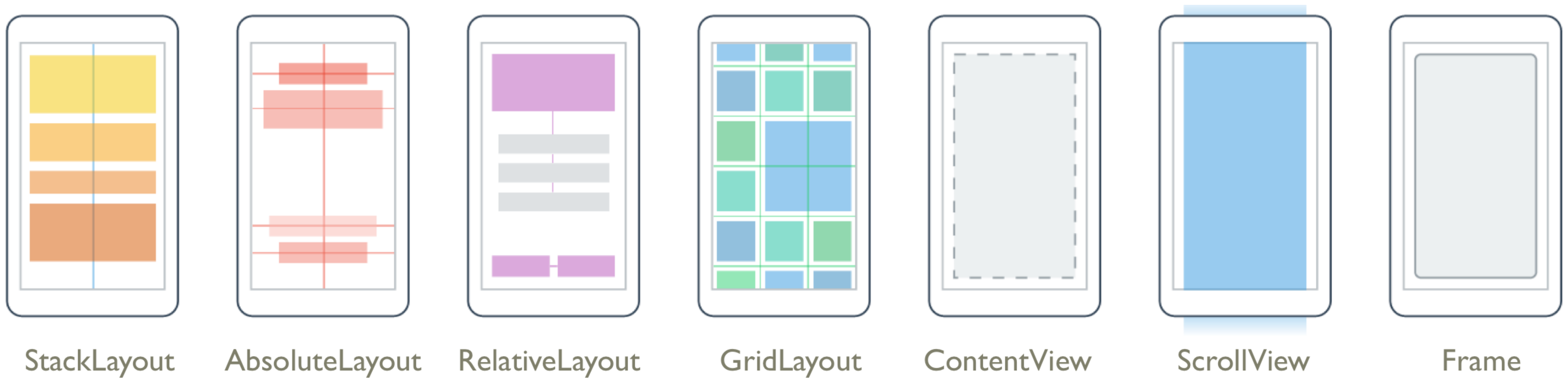
- A page is used to define a single screen that contains most or all of the screen.



@MobileRez

What is a Layout?

- A layout is a special type of view that acts as a container for other views or layouts.



What is a View?

- A View is the term Xamarin.Forms uses for all its basic controls from Buttons to Progress Bars.
- Some of the Views Xamarin.Forms contains are
  - Button
  - Date Picker
  - Entry (*input box*)
  - Label
  - Picker (*The phones form of dropdown list*)
  - Progress Bar

A full list of Views at https://developer.xamarin.com/guides/cross-platform/xamarin-forms/controls/views/

What is a Cell?

- A Cell is a special element that is used inside tables and defines how each item in a list is rendered.
- An example of Cells Xamarin.Forms supports:
  - Entry Cell
  - Switch Cell
  - Text Cell
  - Image Cell

A full list of Cells at https://developer.xamarin.com/guides/cross-platform/xamarin-forms/controls/cells/

Traditional way to build Forms apps

- Xamarin.Forms apps are commonly built using all using C# and not XAML.
- A new Xamarin.Forms app is usually created with a dummy app in a cs file

```
public App()
{
    // The root page of your application
    MainPage = new ContentPage {
        Content = new StackLayout {
            VerticalOptions = LayoutOptions.Center,
            Children = {
                new Label {
                    XAlign = TextAlignment.Center,
                    Text = "Welcome to Xamarin Forms!“
                }
            }
        }
    };
}
```

@MobileRez

Working with XAML Syntax & Behavior

What is XAML?

- XAML stands for E<u>x</u>tensible <u>A</u>pplication <u>M</u>arkup <u>L</u>anguage and was created by Microsoft specifically for working with the UI
- A XAML file is always associated with a C# code file.

Why use XAML over all code in a .cs file?

- Designer can create UI while coder focuses on code in the code file
- XAML allows for features like DataBinding Animations, Custom behaviors, value converters & more.
- Easier to work with for those who like to have a more visual representation of their layouts
- Helps keep a separation between UI and app logic

Building a layout in XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Toolbox.View.Page1">
  <StackLayout>
    <StackLayout.Padding>
      <OnPlatform x:TypeArguments="Thickness“
                  iOS="0, 20, 0, 0"/>
    </StackLayout.Padding>
    <Label Text="{Binding MainText}" VerticalOptions="Center" HorizontalOptions="Center" />
    </StackLayout>
</ContentPage>
```

@MobileRez

XAML Syntax: Attached properties & Property Elements

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="Toolbox.View.MunitConverter“>
  <StackLayout VerticalOptions="CenterAndExpand" Padding="20" >
    <Grid HorizontalOptions="Center" >
      <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="*"/>
      </Grid.RowDefinitions>
      ...
      <Label x:Name="endLabel" Grid.Row="7" StyleId="EndLabel" XAlign="Center"/>
    </Grid>
  </StackLayout>
</ContentPage>
```

@MobileRez

Platform Targeted Tweaks

- Building a cross platform app means you may need to tweak a specific platform or device class (phone or tablet) to make it match the other platforms. To do this we can use OnPlatform and OnIdiom

OnPlatform & OnIdiom

- OnPlatform allows us to define code for a specific platform
- OnIdiom allows us to define code for either a tablet or phone
- Both are useful for providing those quick tweaks to get the UI to look the same on all devices or only changing one platform to fit the rest of their needs
- Can be used with padding, color, size, font, width, height, and strings/text

OnPlatform & OnIdiom In Action

- OnPlatform is used here to add a thickness at the top of the app on iOS to make room for the top menu. OnIdiom Is used to set a color based on the platform.

```
<StackLayout>
  <StackLayout.Padding>
    <OnPlatform x:TypeArguments="Thickness“
                Android="0, 0, 0, 0“
                iOS="0, 20, 0, 0“
                WinPhone="0, 0, 0, 0"/>
    </StackLayout.Padding>
    <StackLayout.BackgroundColor>
      <OnIdiom x:TypeArguments="Color“
               Phone="Teal“
               Tablet="Green"/>
    </StackLayout.BackgroundColor>
    <Label Text="{Binding MainText}" VerticalOptions="Center" HorizontalOptions="Center" />
   </StackLayout>
```

@MobileRez

Advanced XAML

Reducing Duplicated styles with XAML

- When defining multiple and app wide styles, we run into the issue of needing to get the most out of our code and limit reuse or duplication of similar styles. In XAML we can do this with Resource Dictionaries.

Resource Dictionary's

- A Resource Dictionary Is a dictionary that can be defined in the XAML or code behind and is used to hold UI resources such as colors, fonts, and styles.
- A resource in the Dictionary Uses the x:Key to define the ID of that resource so you can reference it later.

Using a Resource Dictionary

- In the example below, we use the resource dictionary to define colors for specific platforms and values that we plan to reuse like centering

```
<ContentPage.Resources>
  <ResourceDictionary>
    <Color x:Key="TxtRed">Red</Color>
    <OnPlatform x:Key="TxtColor“ x:TypeArguments="Color“ Android="Green“ iOS="Teal“
                WinPhone="Purple"/>
    <LayoutOptions x:Key="HorzCenter">Center</LayoutOptions>
  </ResourceDictionary>
</ContentPage.Resources>
<StackLayout>
  <Label Text="Im a label" HorizontalOptions="{StaticResource HorzCenter}“
         TextColor="{StaticResource TxtColor}" />
  <Label Text="Im a label too, but different" HorizontalOptions="{StaticResource HorzCenter}“
         TextColor="{StaticResource TxtRed}"/>
  <Label Text="Im also a different label but look like the first label"
         HorizontalOptions="{StaticResource HorzCenter}“
         TextColor="{StaticResource TxtColor}"/>
</StackLayout>
```

@MobileRez

Resource Dictionary hierarchy

- Resource files can inherit from global resource files
- Resource files prioritize definitions closer to where they started in the hierarchy.
- Order of priority is View, Layout, Page, Application

Resource Dictionary Hierarchy

App.xaml

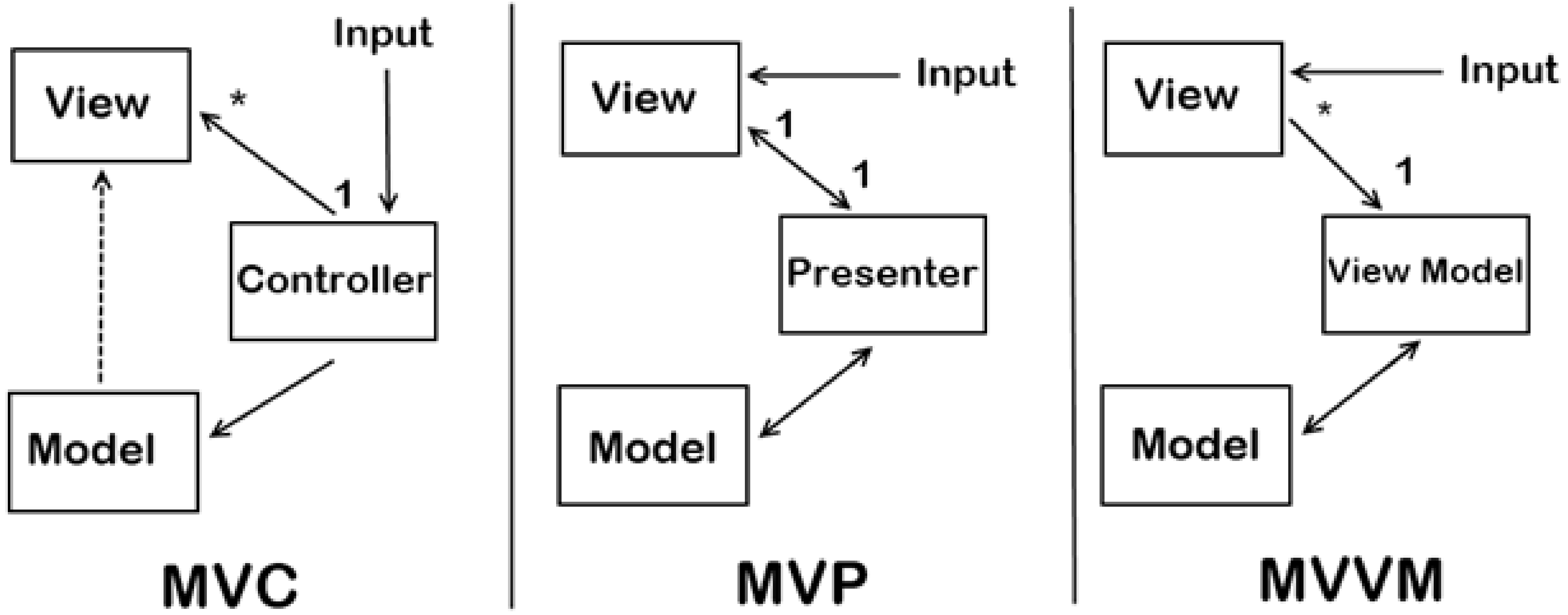
```
  <ContentPage.Resources>
     <ResourceDictionary>
       <x:String x:Key="TxtL1">I’m different</x:String>
     </ResourceDictionary>
   </ContentPage.Resources>
```

Page1.xaml

```
  <ContentPage.Resources>
     <ResourceDictionary>
       <x:String x:Key="TxtL1">Im a label</x:String>
     </ResourceDictionary>
   </ContentPage.Resources>
   <StackLayout>
     <Label Text="{StaticResource TxtL1}"></Label>
   </StackLayout>
```

MVVM (Model-View-ViewModel)

- MVVM is a layer separated method of coding and organizing files commonly found when using XAML.



@MobileRez

Why Use Data Binding with Xamarin.Forms?

- Data Binding creates a one or two way relationship between the source and target properties
- Must be used with a bindable property

Hands on Lab

What's Next?

- Data Binding with Xamarin.Forms & XAML
- List views and collections with Data Binding, XAML, & Xamarin.Forms

Questions?

mitchmuenster@gmail.com

@MobileRez

mitchmuenster.com