

# Getting hands on with XAML and Xamarin.Forms



Mitch "Rez" Muenster

@MobileRez

[mobilerez.tumblr.com](http://mobilerez.tumblr.com)

Xamarin Certified Developer

# Objectives

- ▶ Understanding Xamarin.Forms
- ▶ XAML Syntax & Behavior
- ▶ Advanced XAML

# Understanding Xamarin.Forms

# What is Xamarin.Forms?

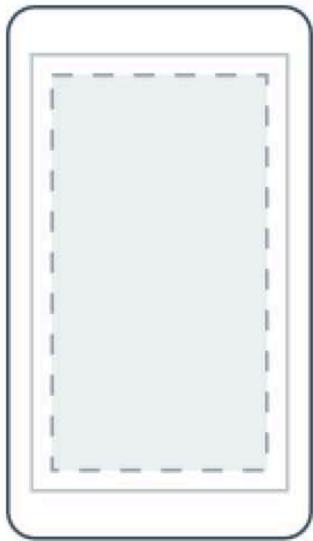
- ❖ Xamarin.Forms allows you to rapidly create a cross platform app with a native UI.
- ❖ Can be created in either a Shared Class Library or Portable Class Library
- ❖ Great for Prototyping or Data-Driven apps.
- ❖ Can also use dependency service to access platform specific features.

# Understanding Xamarin.Forms UI

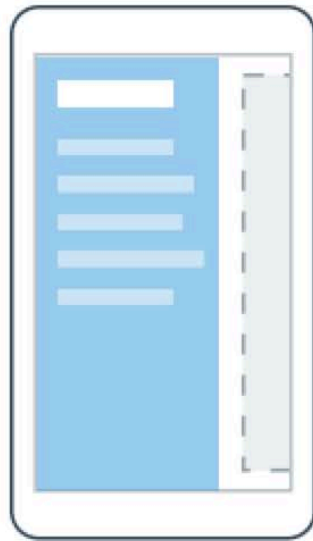
- ❖ Xamarin.Forms UI is defined in 4 different ways; Pages, Layouts, Cells, and Views.

# What is a Page?

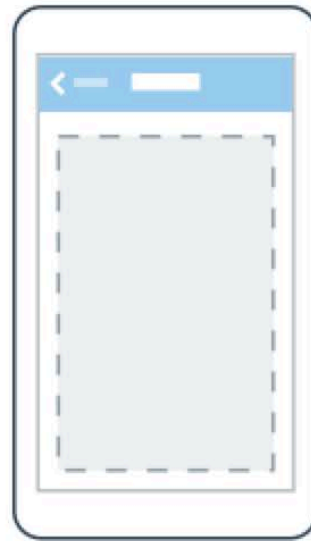
- ❖ A page is used to define a single screen that contains most or all of the screen.



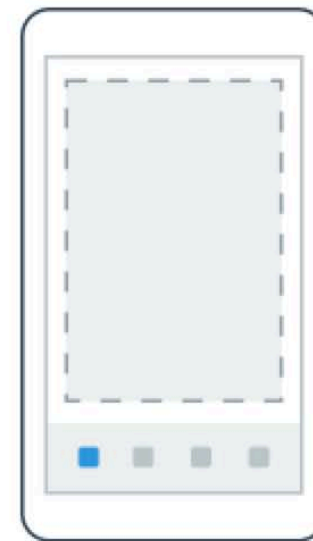
ContentPage



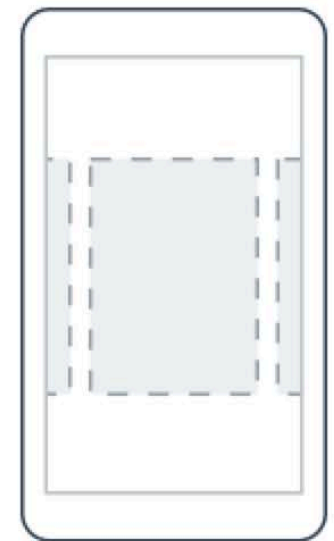
MasterDetailPage



NavigationPage



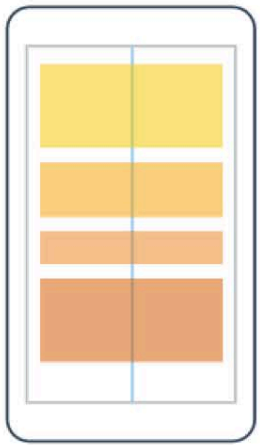
TabbedPage



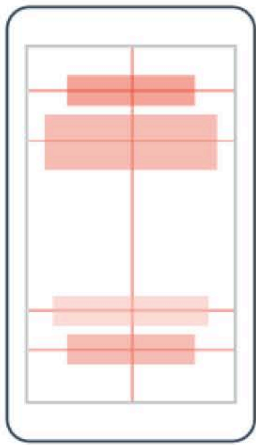
CarouselPage

# What is a Layout?

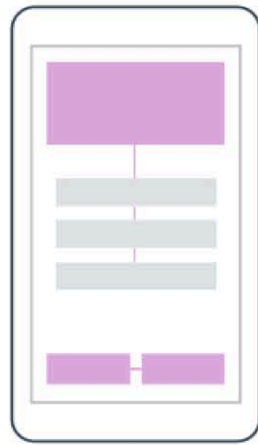
- ❖ A layout is a special type of view that acts as a container for other views or layouts.



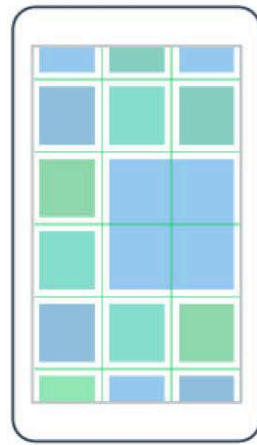
StackLayout



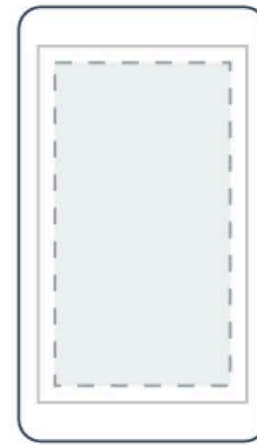
AbsoluteLayout



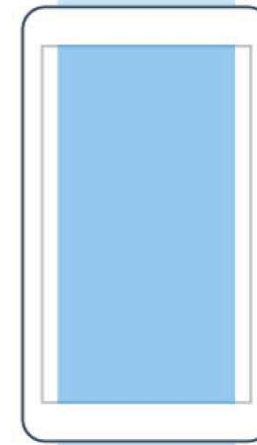
RelativeLayout



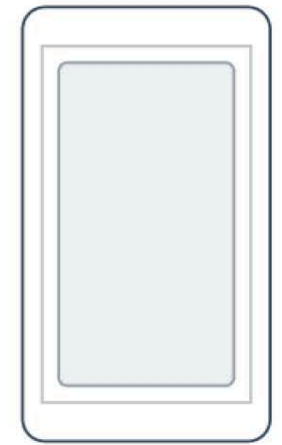
GridLayout



ContentView



ScrollView



Frame

# 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!"
                }
            }
        }
    };
}
```

# XAML Syntax & Behavior

# What is XAML?

- ❖ XAML stands for Extensible Markup Language 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

# Breakdown Of a XAML File

# XAML Syntax

# Building a layout in XAML



# Using OnPlatform

OnPlatform is used to specify platform-specific logic for a view or view model.

It is a property of the `View` or `ViewModel` class that takes a `Platform` enum and a delegate.

For example, you can use `OnPlatform` to specify different colors for different platforms:

```
public class MyView : View  
{  
    public MyView() : base()  
    {  
        BackgroundColor = OnPlatform(Platform.Android, Color.Red, Color.Blue);  
    }  
}
```

Or you can use it to specify different actions for different platforms:

```
public class MyViewModel : ViewModel  
{  
    public MyViewModel()  
    {  
        ClickCommand = OnPlatform(Platform.Android, new Command(), new Command());  
    }  
}
```

For more information, see the [OnPlatform](#) documentation page.

# Attached properties

• Attached properties are properties that are attached to an object, but are not part of the object's type.

• They are used to store data that is associated with an object, but is not part of the object's type.

• They are used to store data that is associated with an object, but is not part of the object's type.

• They are used to store data that is associated with an object, but is not part of the object's type.

• They are used to store data that is associated with an object, but is not part of the object's type.

• They are used to store data that is associated with an object, but is not part of the object's type.

• They are used to store data that is associated with an object, but is not part of the object's type.

# Advanced XAML

# Using Resource Dictionary

# Resource Dictionary hierarchy

# Data Binding + XAML + Forms

# 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](mailto:mitchmuenster@gmail.com)

[@MobileRez](#)

[mobilerez.tumblr.com](http://mobilerez.tumblr.com)