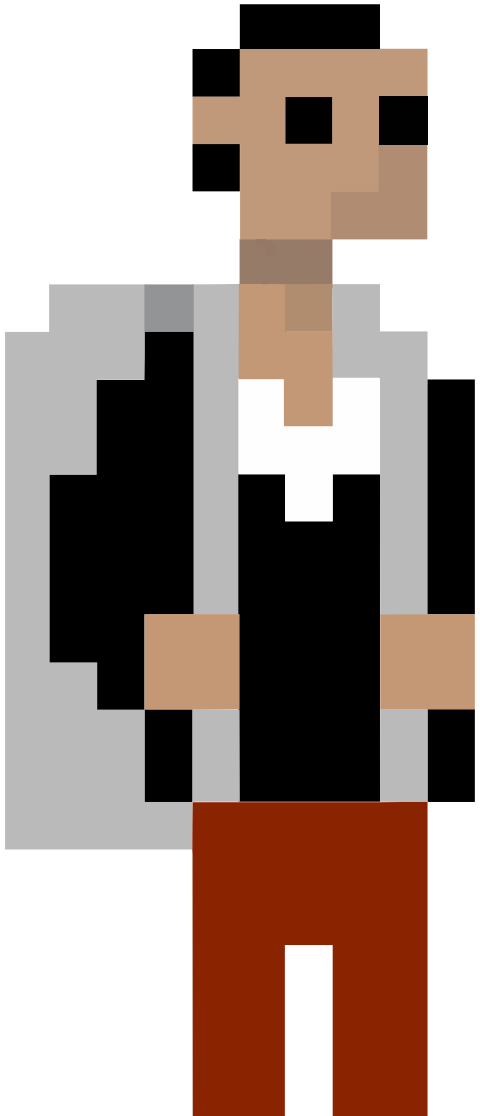
Spice Up Your Xamarin.Forms UIs with Custom Renderers

Pierce Boggan

Who’s this guy?



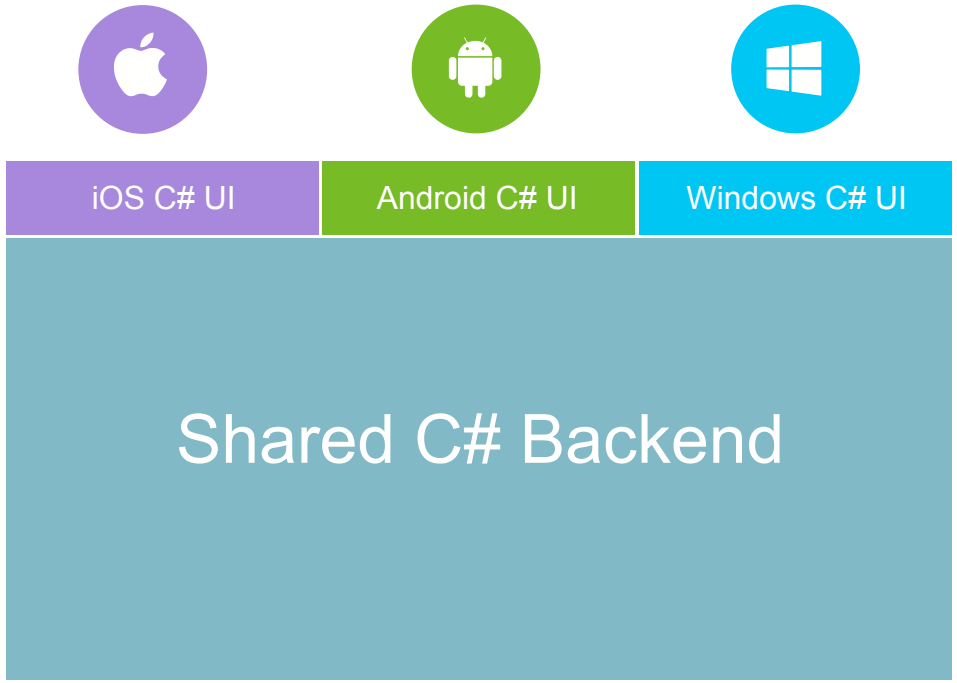
Pierce Boggan
Developer Evangelism Team, Xamarin

pierce@xamarin.com
@pierceboggan

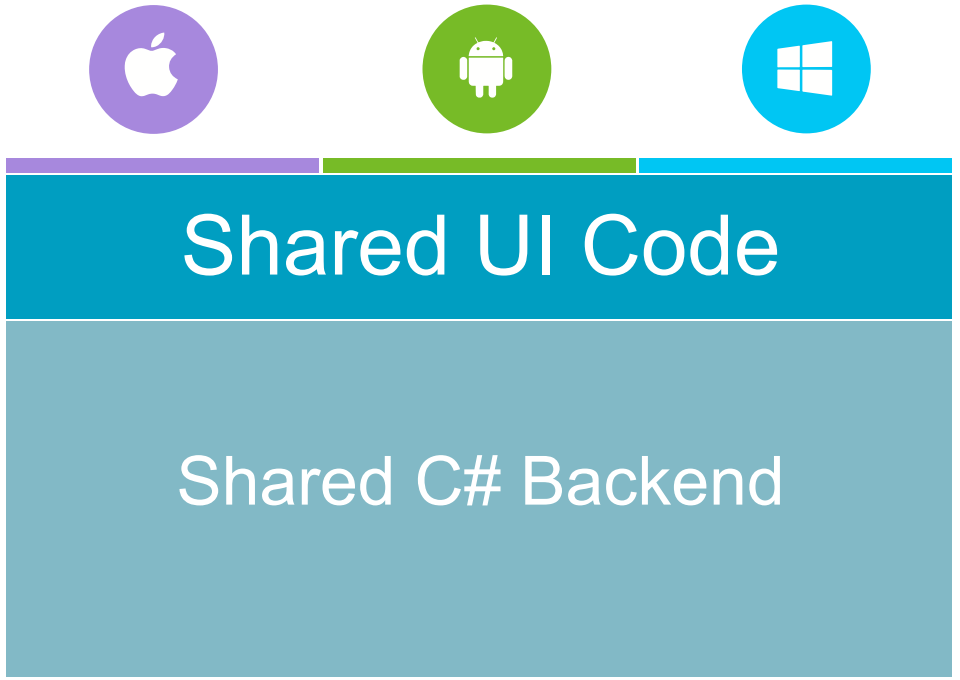
Meet Xamarin.Forms

Build native UIs for iOS, Android, and Windows Phone from a single, shared C# codebase.

Xamarin + Xamarin.Forms

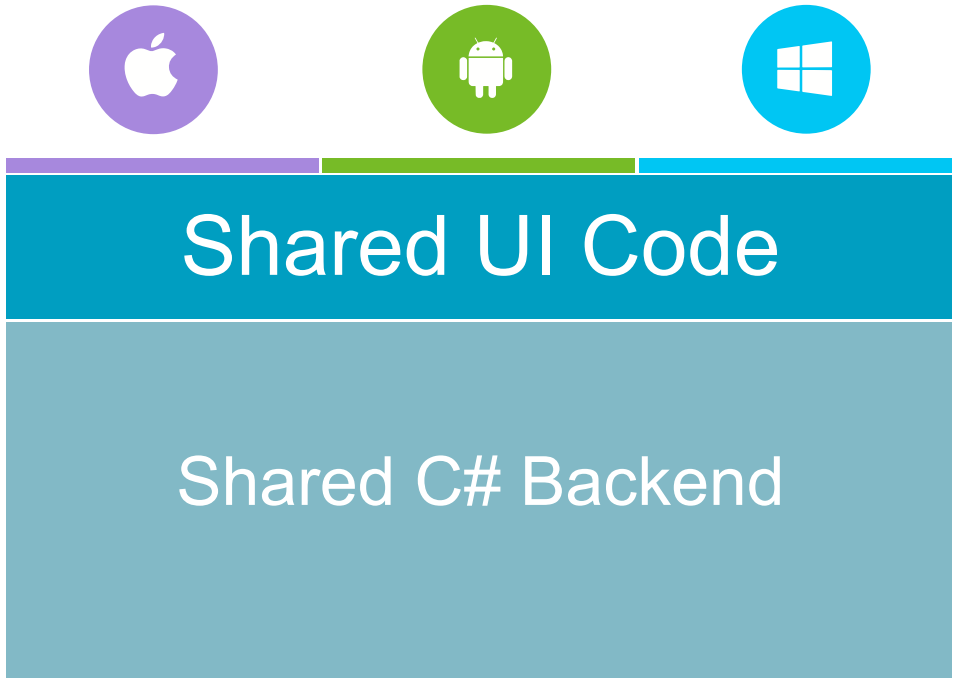


Traditional Xamarin Approach



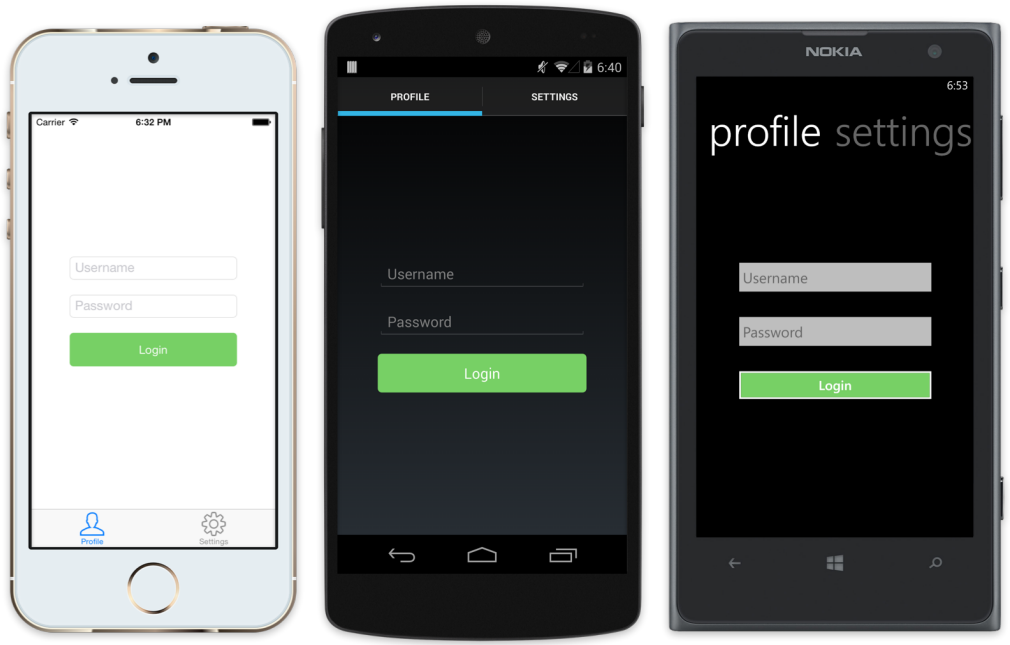
With Xamarin.Forms:
More code-sharing, all native

What's included



- ✓ 40+ Pages, layouts, and controls
  - (Build from code behind or XAML)
- ✓ Two-way data binding
- ✓ Navigation
- ✓ Animation API
- ✓ Dependency Service
- ✓ Messaging Center

Native UI from shared code




```
<?xml version="1.0" encoding="UTF-8"?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="MyApp.MainPage">
<TabbedPage.Children>
<ContentPage Title="Profile" Icon="Profile.png">
    <StackLayout Spacing="20" Padding="20"
                 VerticalOptions="Center">
        <Entry Placeholder="Username"
               Text="{Binding Username}"/>
        <Entry Placeholder="Password"
               Text="{Binding Password}"
               IsPassword="true"/>
        <Button Text="Login" TextColor="White"
                BackgroundColor="#77D065"
                Command="{Binding LoginCommand}"/>
    </StackLayout>
</ContentPage>
<ContentPage Title="Settings" Icon="Settings.png">
    <!-- Settings -->
</ContentPage>
</TabbedPage.Children>
```

Which **Xamarin approach** is best for your app?

Xamarin.Forms is best for:

- Data entry apps
- Prototypes and proofs-of-concept
- Apps that require little platform-specific functionality
- Apps where code sharing is more important than custom UI

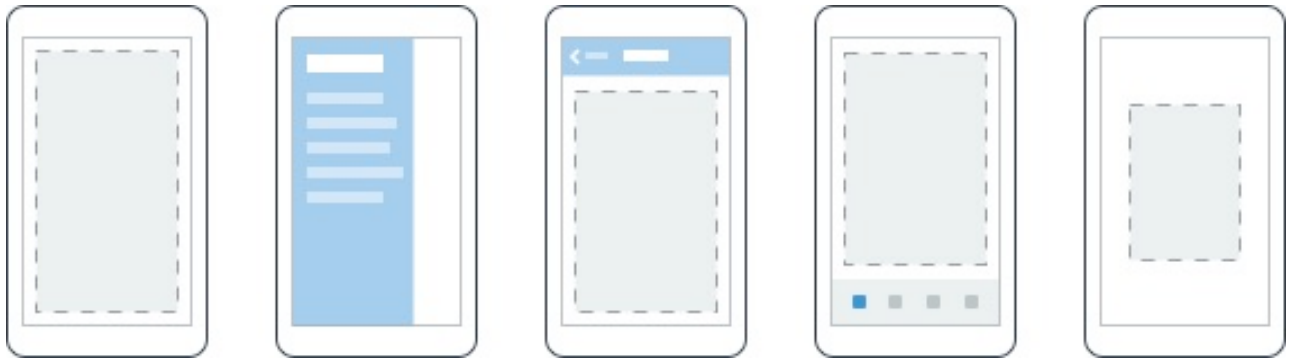
Learn more: xamarin.com/forms

Xamarin.iOS / Xamarin.Android is best for:

- Apps that require specialized interaction
- Apps with highly polished design
- Apps that use many platform-specific APIs
- Apps where custom UI is more important than code sharing

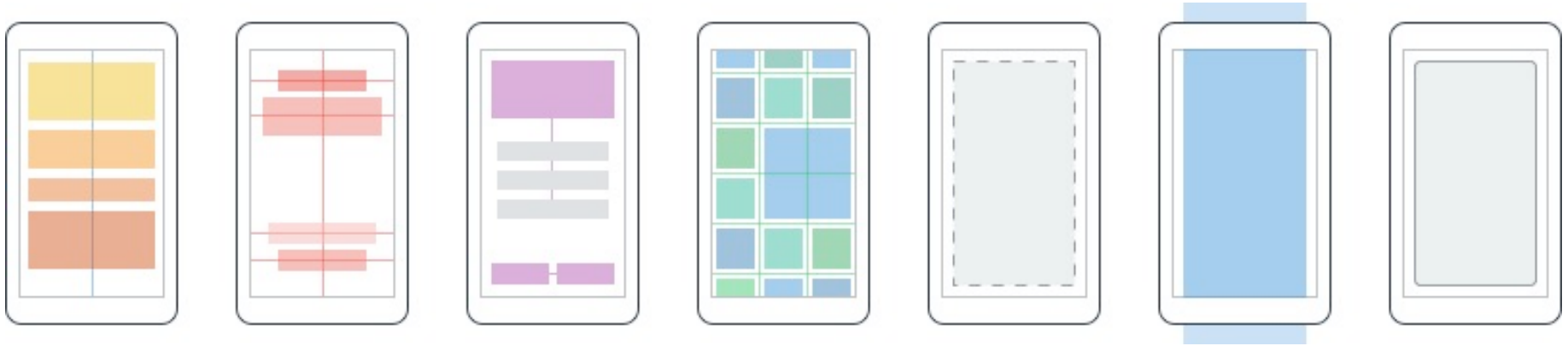
Learn more: xamarin.com/platform

Pages



Content MasterDetail Navigation Tabbed Carousel

Layouts



Stack Absolute Relative Grid ContentView ScrollView Frame

Controls

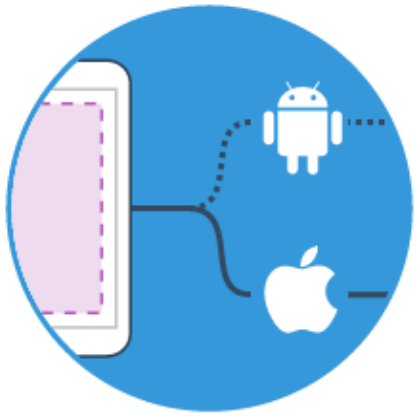
ActivityIndicator	BoxView	Button	DatePicker	Editor
Entry	Image	Label	ListView	Map
OpenGLView	Picker	ProgressBar	SearchBar	Slider
Stepper	TableView	TimePicker	WebView	EntryCell
ImageCell	SwitchCell	TextCell	ViewCell	

Xamarin.Forms Ecosystem

Extensibility

Embed Custom Views Anywhere

Not only are Xamarin.Forms pages mixable with custom screens, you can embed custom views built directly against Xamarin.iOS and Xamarin.Android into Xamarin.Forms pages.



Call Platform APIs via Shared Services

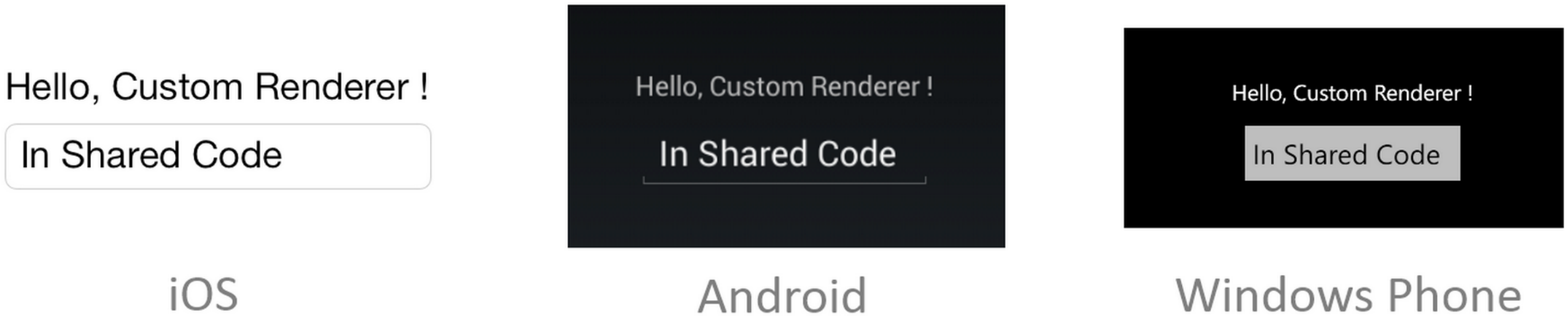
Need platform-specific functionality within a Xamarin.Forms page, such as querying the accelerometer? We've included services that quickly and easily abstract platform-specific API calls to keep you writing as much shared code as possible.

Easy migration to the Xamarin Platform

When your native platform integration requirements outgrow Xamarin.Forms, your shared code is easily migrated to Xamarin.iOS and Xamarin.Android using the full Xamarin platform.

Exploring Custom Renderers

User interfaces built with Xamarin.Forms look and feel native because they *are* native.



The “Entry” control rendered on three platforms natively.

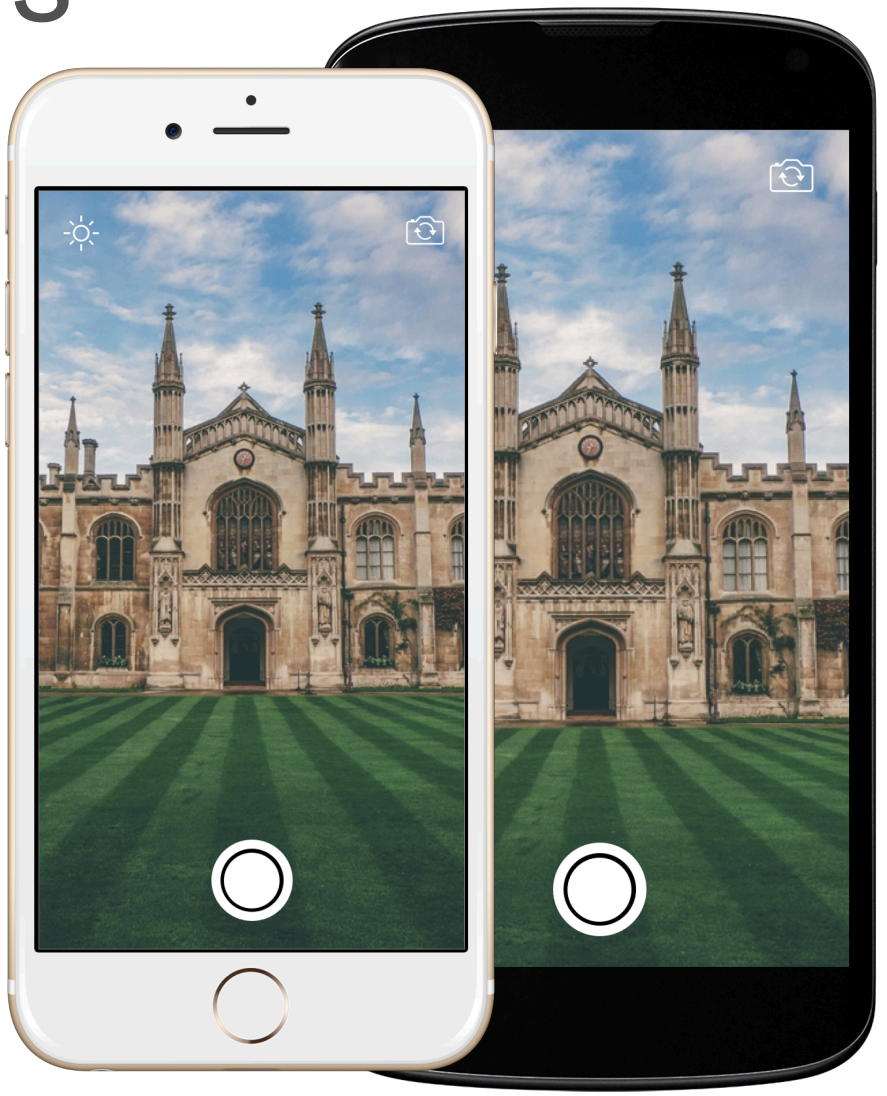
Exploring Custom Renderers

- Pages, layouts, and controls represent a common API to describe cross-platform user interfaces
- Each control is rendered differently on each platform
- A *Renderer* class creates a rendering for a Xamarin.Forms control using native controls (such as *UITextField*, *EditText*, and *TextBox*)

Exploring Custom Renderers

By tapping into the built-in rendering engine, developers can:

- ✓ Modify existing controls
- ✓ Build custom controls
- ✓ Create custom pages

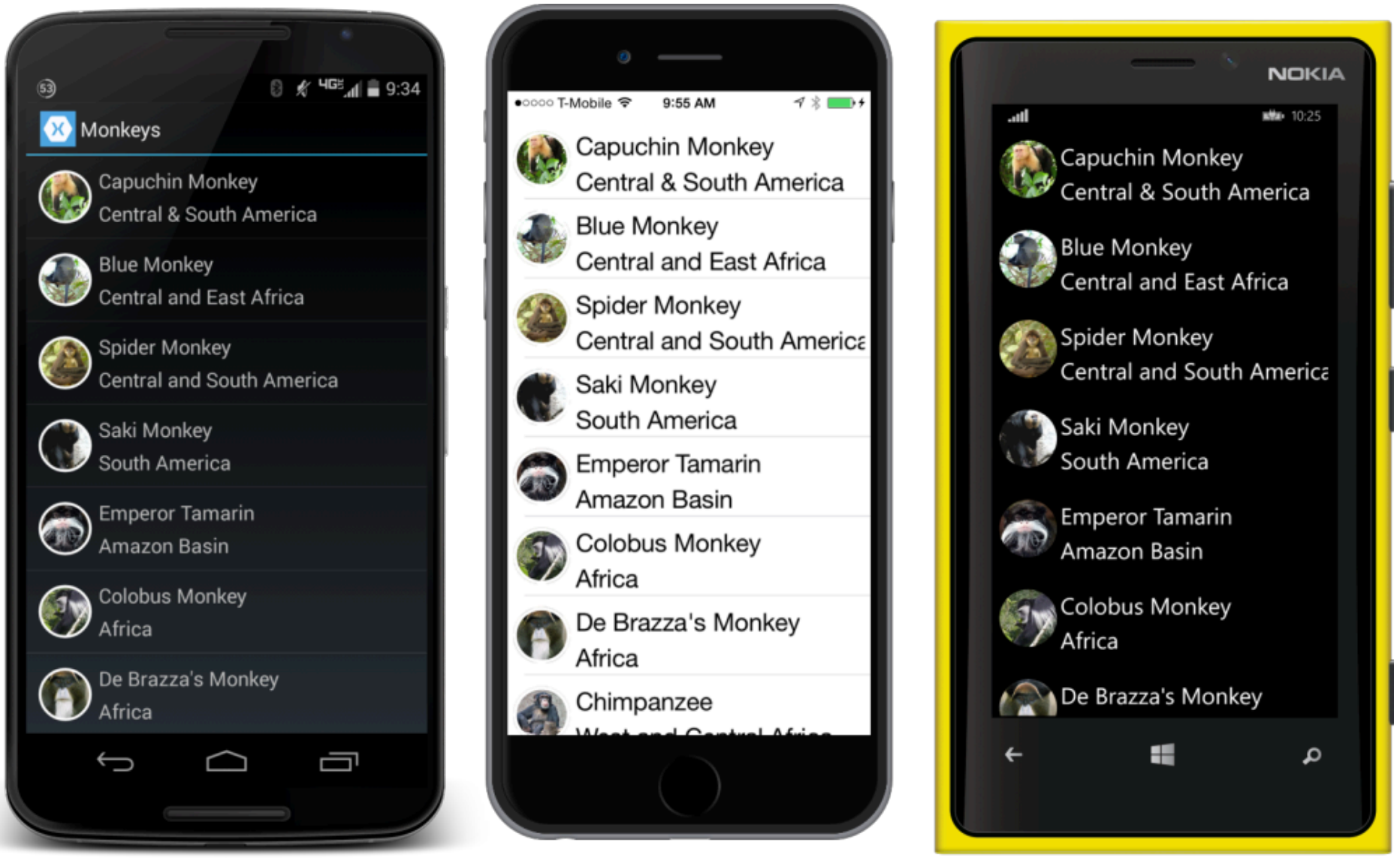


Moments

Moments

- Snapchat clone built with Xamarin.Forms
- 87% shared code
- Custom renderers
  - Modified controls
  - Custom control
  - Custom pages

Custom Renderers in the Wild



Custom Renderers in the Wild

Implementing a Custom Renderer

1. Create a new Xamarin.Forms control.
   - Derive from an existing control or the *View* class for new controls.
2. Create a view renderer for each platform targeted.
   - Use an existing renderer class (i.e. *EntryRenderer* or *PageRenderer*)
   - Use *ViewRenderer<TView, TNative>* for new controls.
3. Associate the renderer with the control through the *[ExportRenderer]* attribute.

Modifying Existing Controls Demo

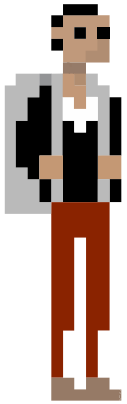
Modifying Existing Controls (Correctly) Demo

Creating Custom Pages Demo

Resources

- Official Documentation
  - Xamarin Blog
  - Custom Renderers Documentation
  - Mark Smith's "Customizing Control Rendering in Xamarin.Forms" Talk
  - *Creating Mobile Apps with Xamarin.Forms*
- Additional Resources
  - Slides & Code
  - Xamarin.Forms Labs

Thank you.
Questions?



Pierce Boggan
Developer Evangelism Team,
Xamarin

pierce@xamarin.com

@pierceboggan