Introduction to Xamarin.Forms

Jonathan Dick

Team Xamarin

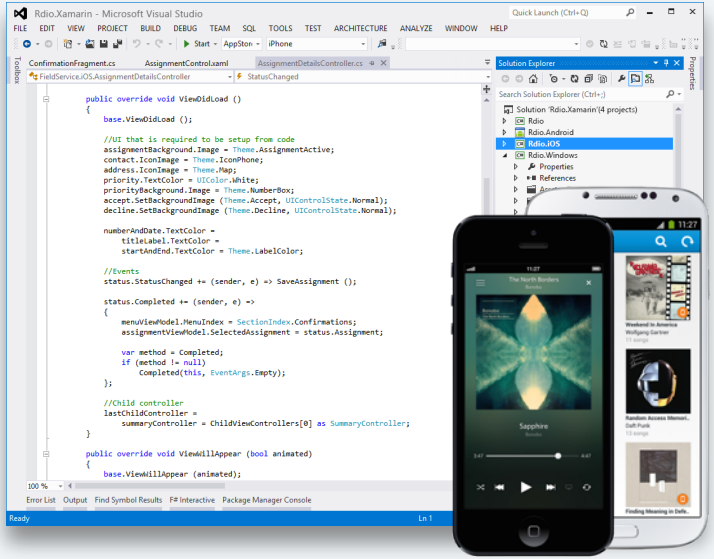
@redth

http://redth.codes

Gone Mobile Podcast

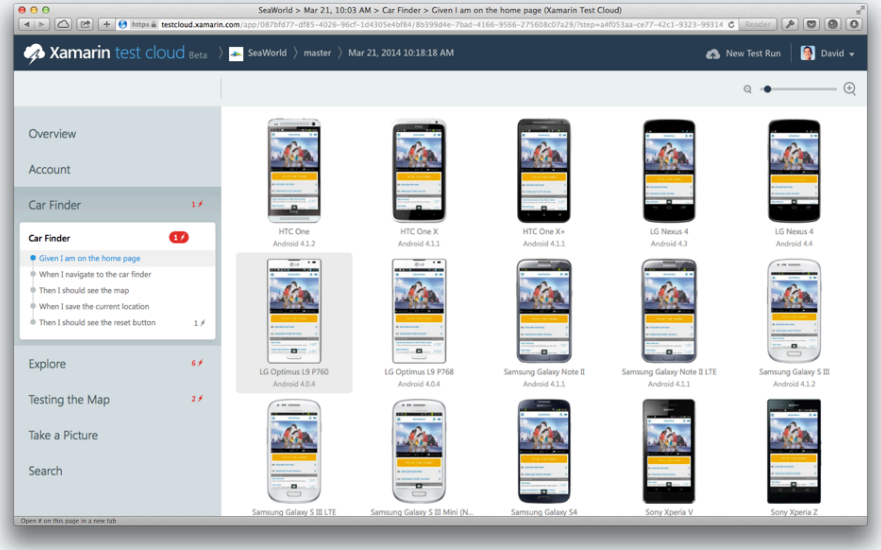
gonemobile.io

Xamarin Platform



Create native iOS, Android, Mac and Windows apps in Visual Studio and C#

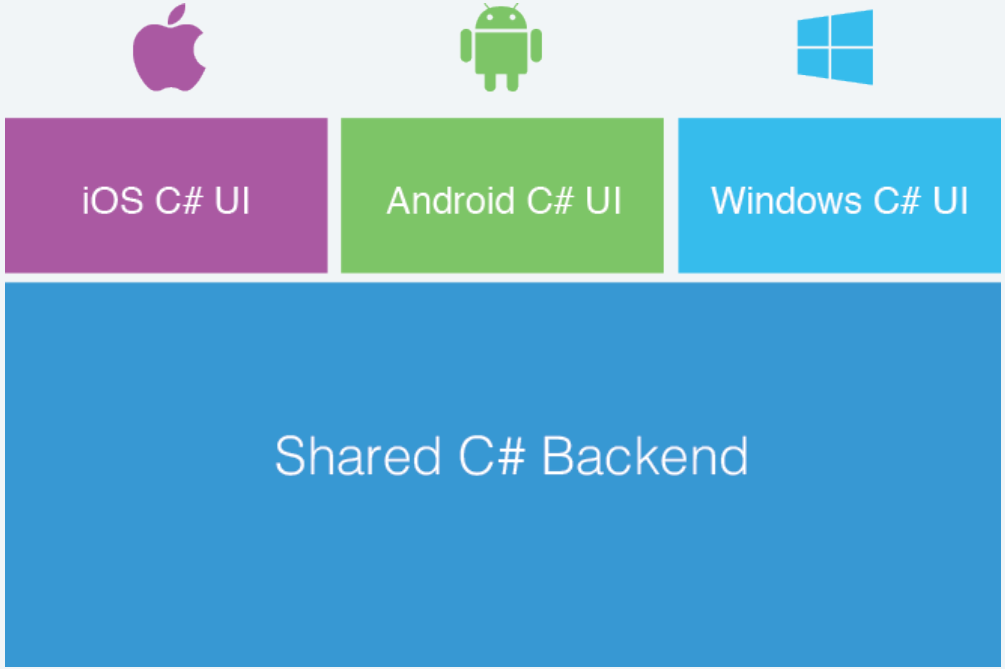
Xamarin test cloud



Automatically test your app on hundreds of mobile devices

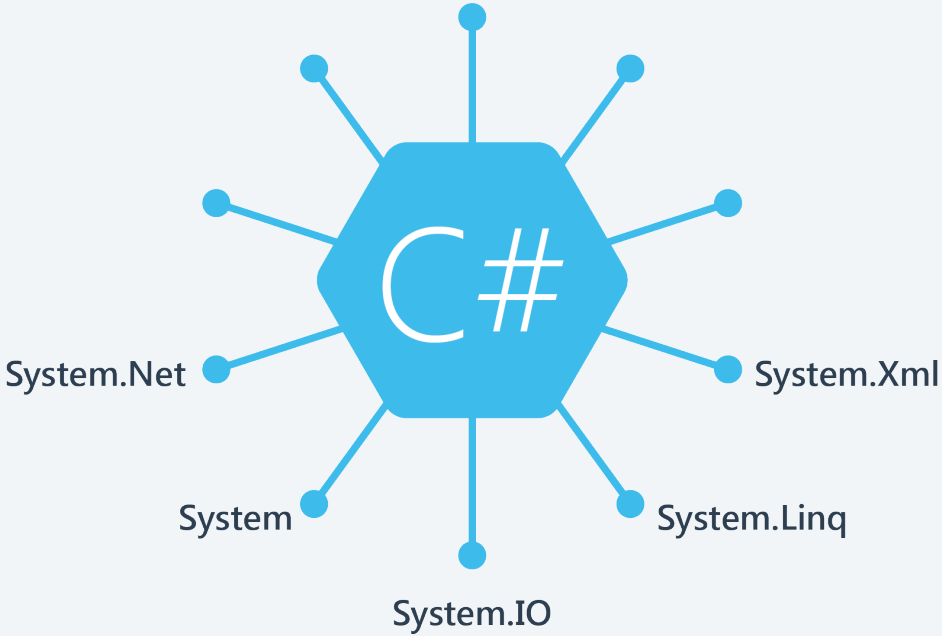
Xamarin's Unique Approach

Native With
Code Sharing



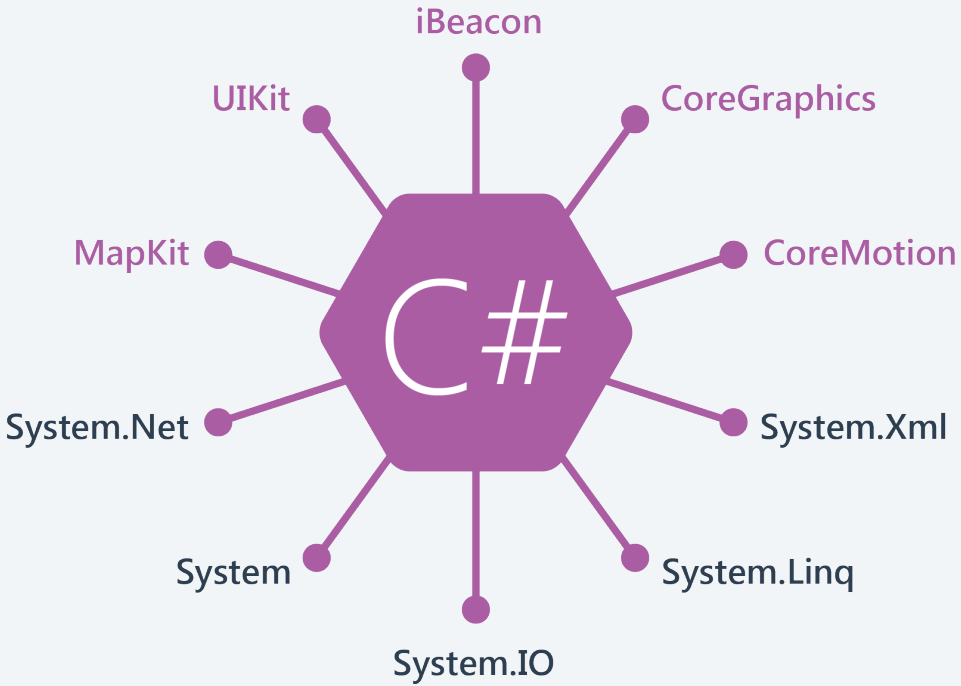
Xamarin

Windows APIs



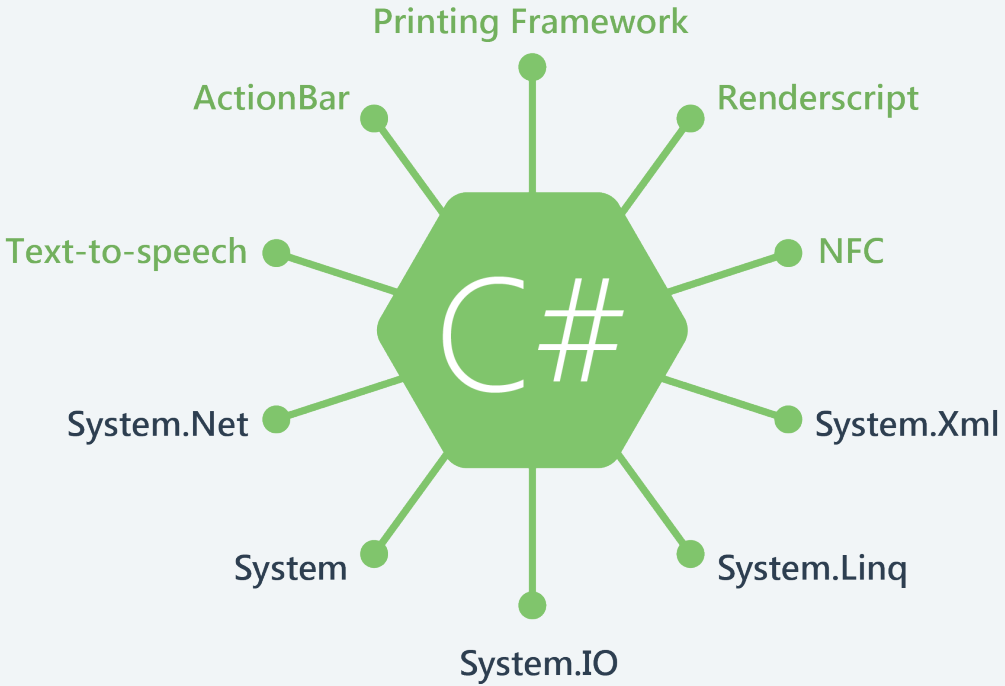
Xamarin

iOS APIs | 100% Coverage



Xamarin

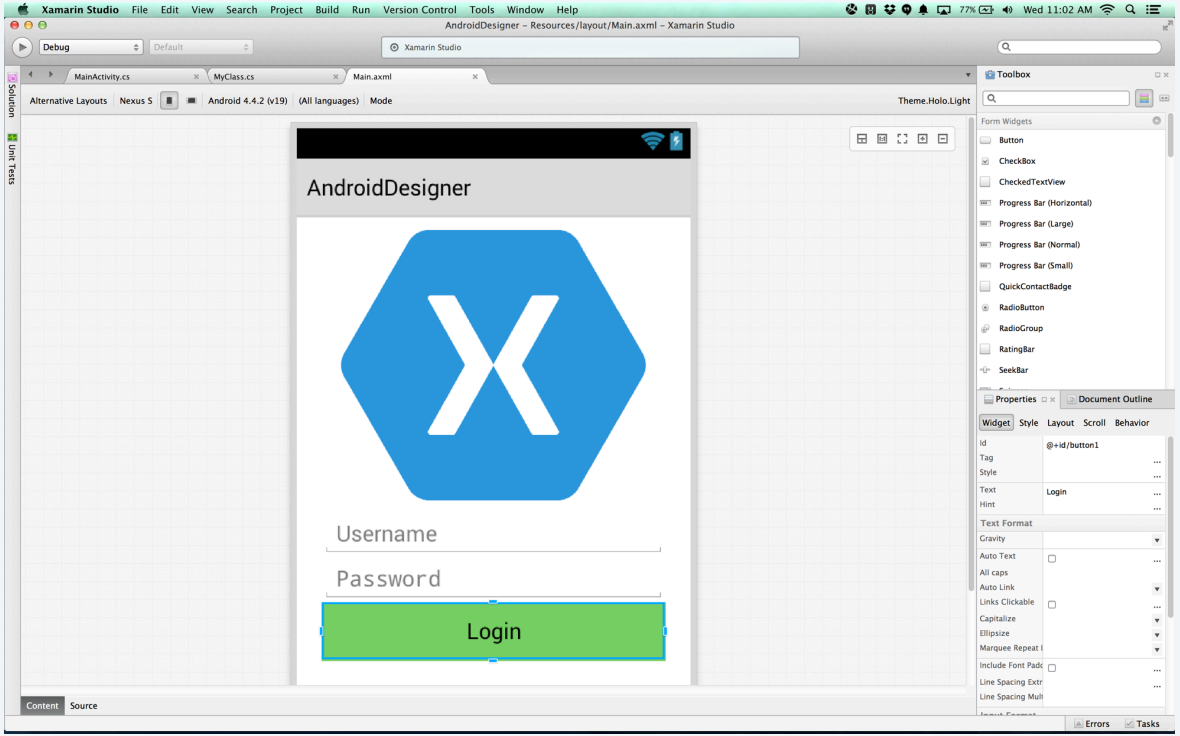
Android APIs | 100% Coverage



Xamarin

Anything you can do in Objective-C, Swift, or Java can be done in C# with Xamarin using Visual Studio

Xamarin Designer for Android



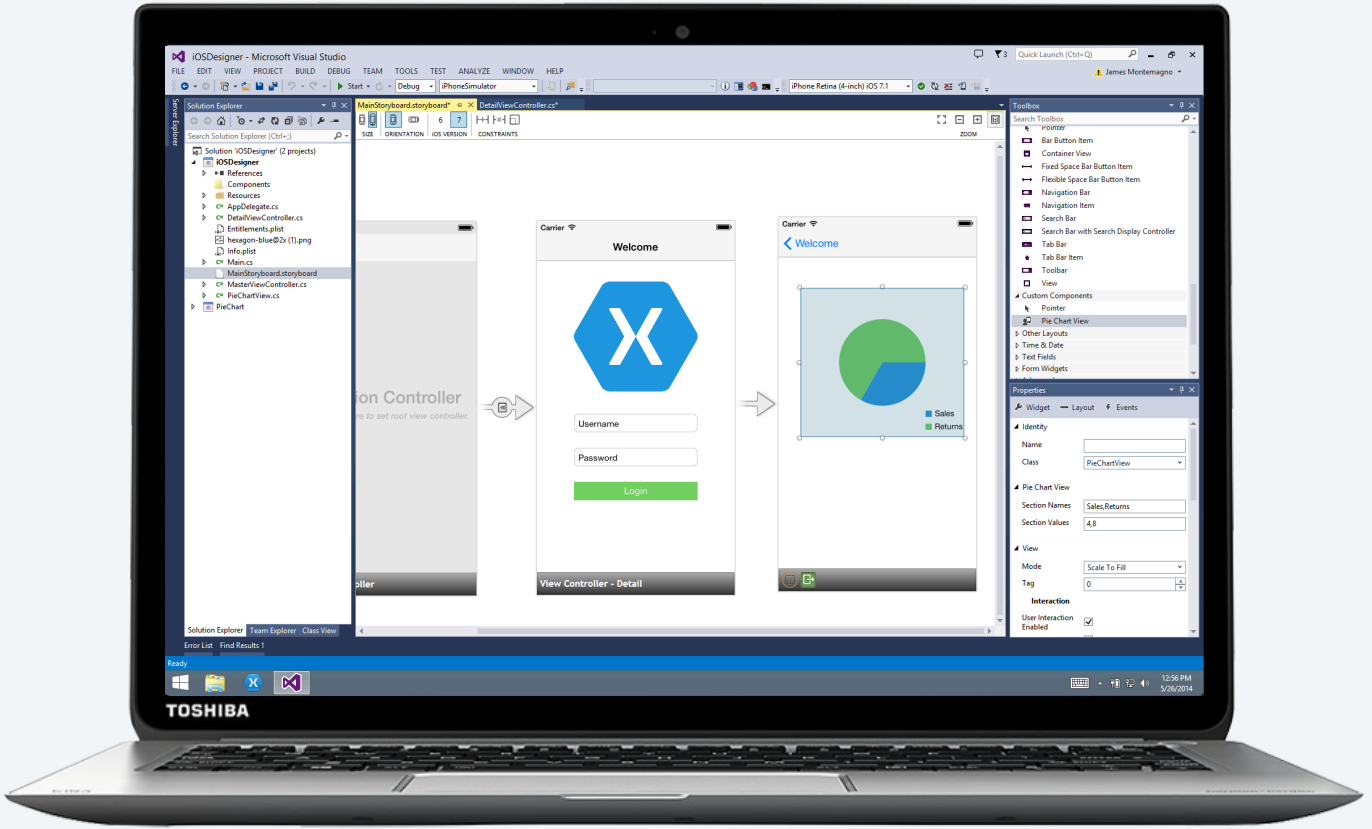
Fully integrated into Xamarin Studio & Visual Studio

Multi-resolution editing

Easy switch between design and Android XML

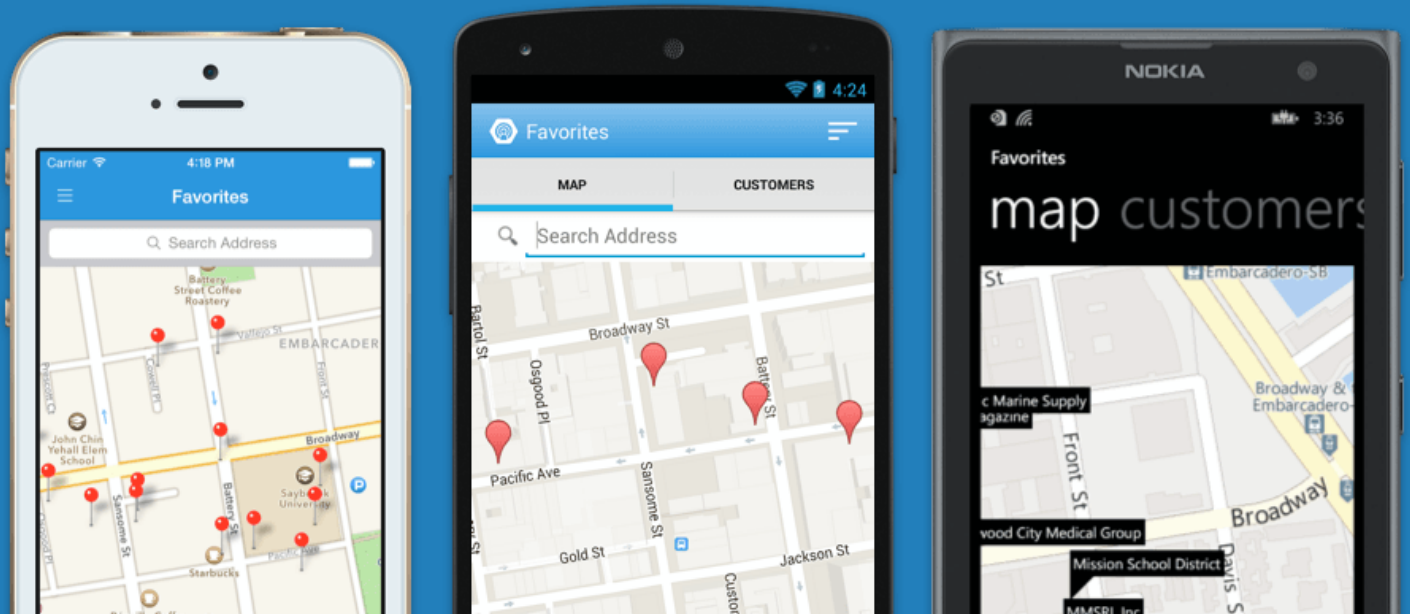
Shipping for over 2 years

Xamarin Designer for iOS



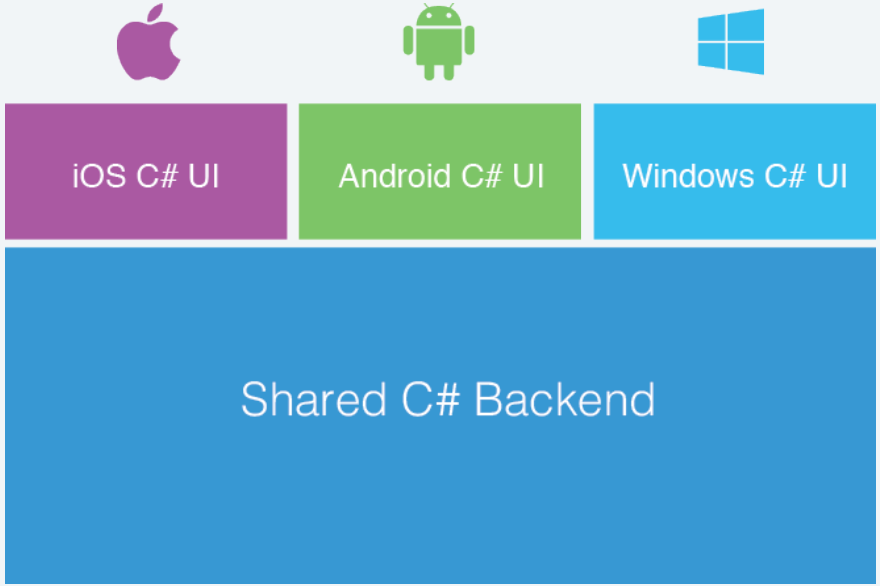
Meet Xamarin.Forms

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

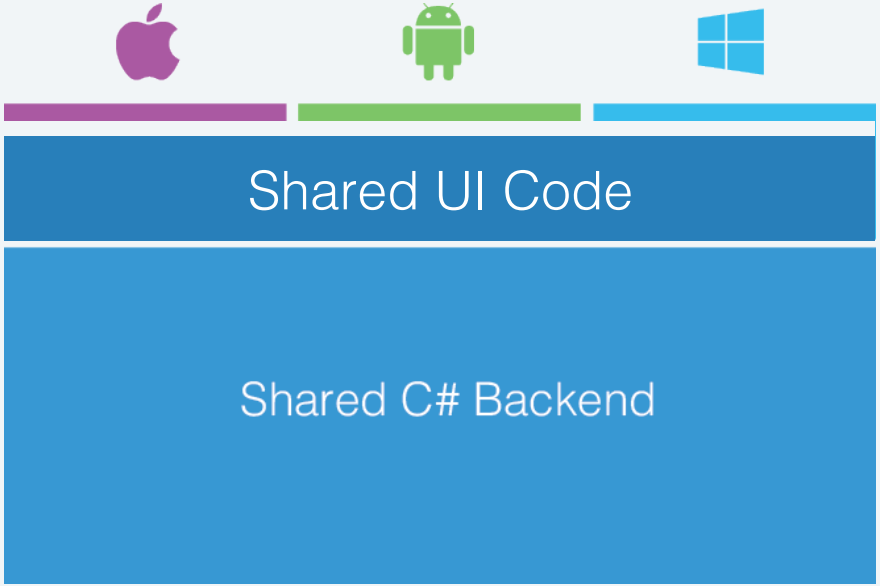


Xamarin + Xamarin.Forms

Traditional Xamarin approach

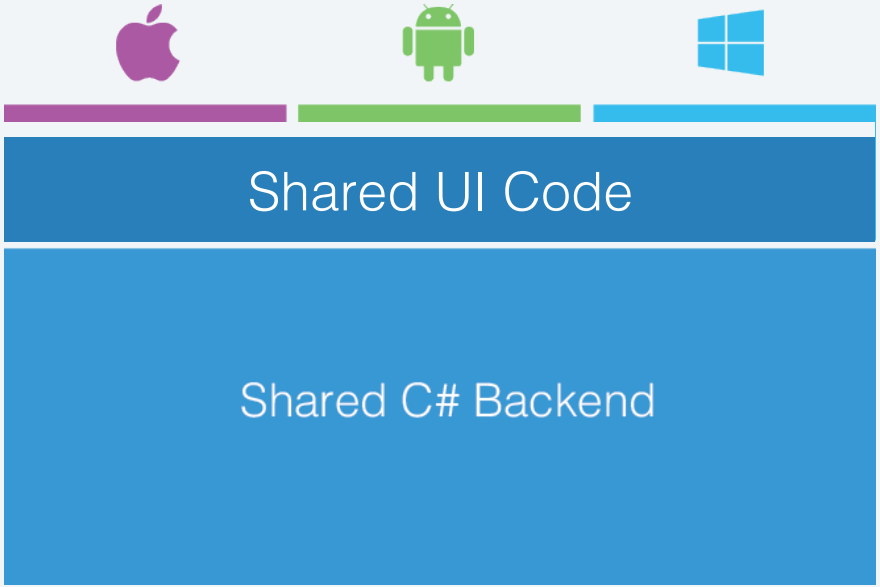


With Xamarin.Forms:
more code-sharing, native controls

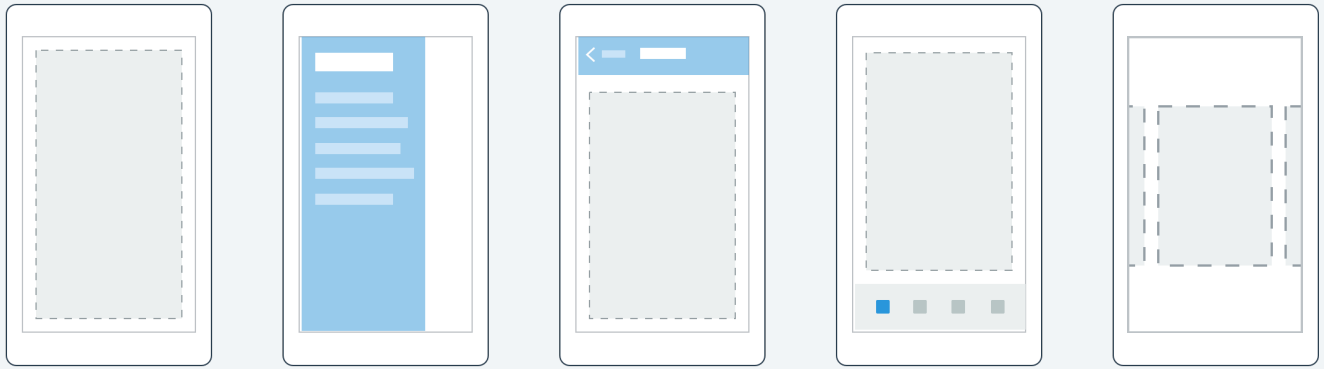


What's Included

- 40+ Pages, Layouts, and Controls
  - Build from code behind or XAML
- Two-way Data Binding
- Navigation
- Animation API
- Dependency Service
- Messaging Center

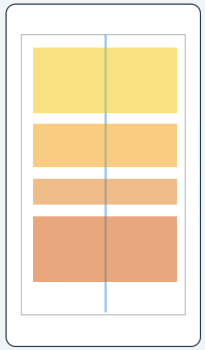


Pages

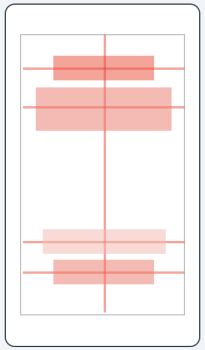


Content MasterDetail Navigation Tabbed Carousel

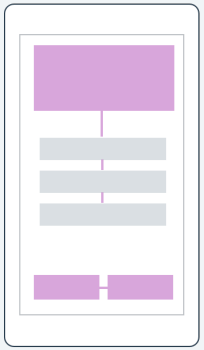
Layouts



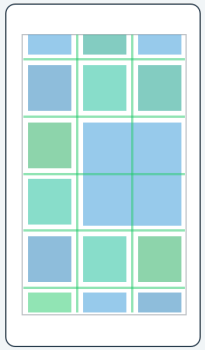
Stack



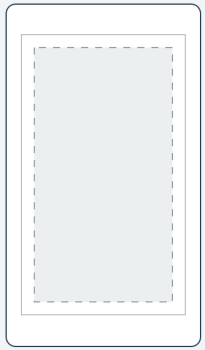
Absolute



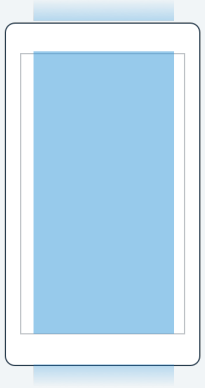
Relative



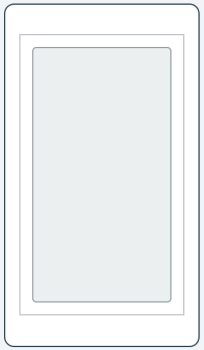
Grid



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	

DEMO 1
Displaying List of Data

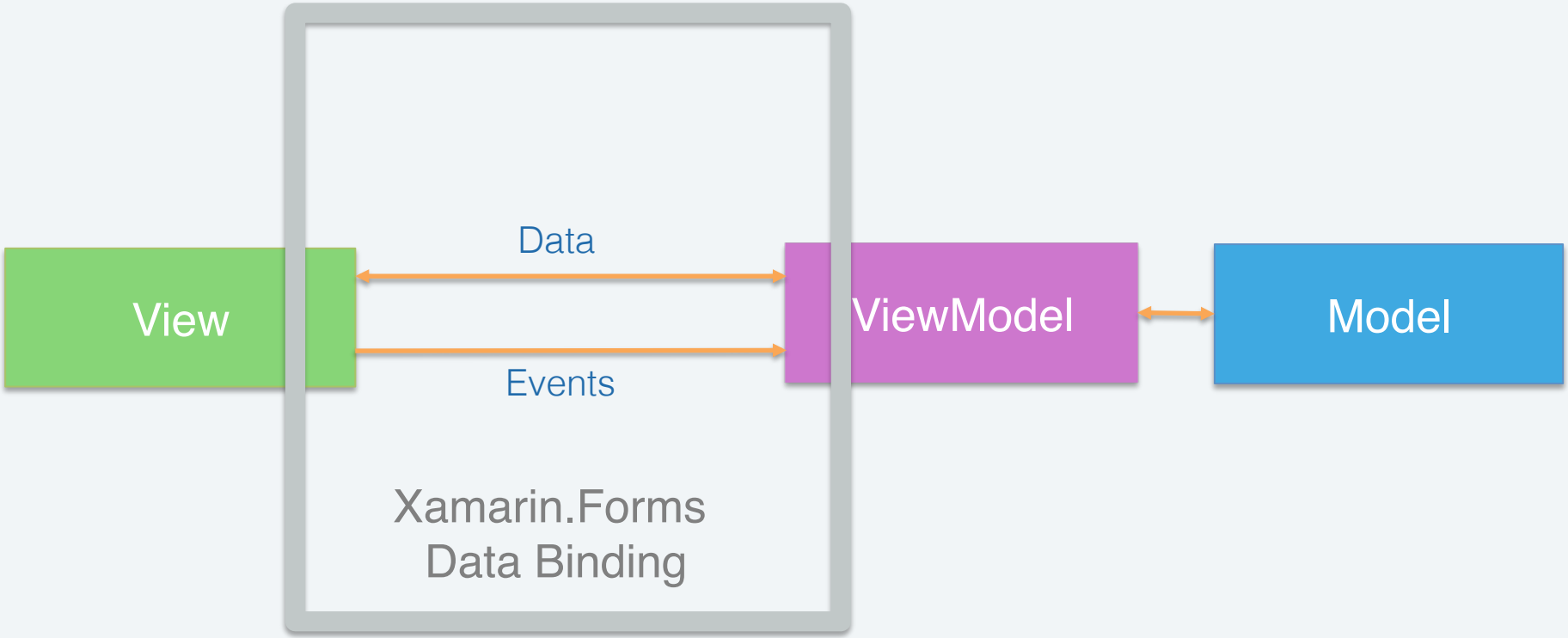
Model-View-ViewModel

How to display information

What to display
Flow of interaction

Business Logic
Data objects

Model-View-ViewModel



Data Binding

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

Data Binding

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

Data Binding

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

```
Label firstName = new Label ();
firstName.SetBinding (Label.TextProperty, "FirstName");
```

Data Binding

```
public class UserViewModel : INotifyPropertyChanged
{
  #region INotifyPropertyChanged implementation
  public event PropertyChangedEventHandler PropertyChanged;
  #endregion

  public void OnPropertyChanged(string name)
  {
    if (PropertyChanged == null)
      return;

    PropertyChanged (this, new PropertyChangedEventArgs (name));
  }
```

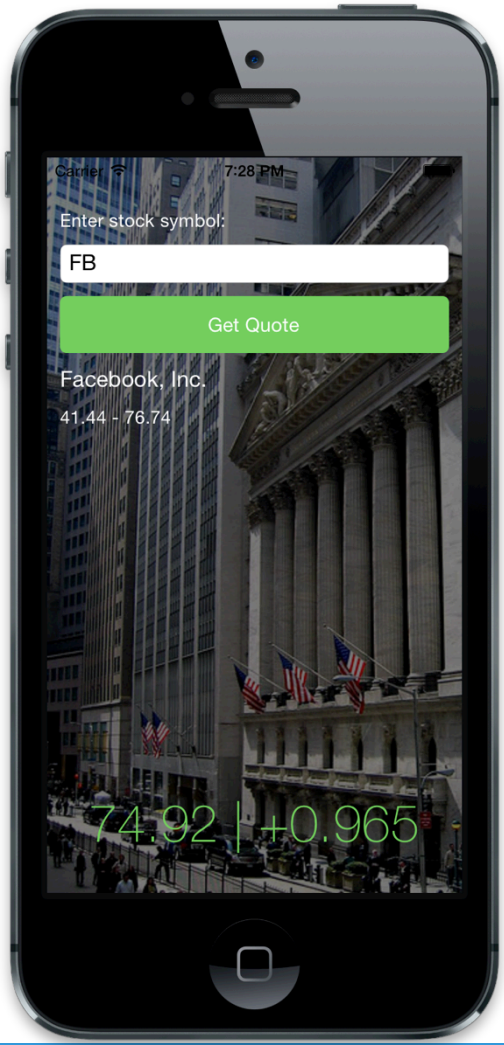
```
private string firstname = string.Empty;
public string FirstName
{
  get { return firstname; }
  set {
    if (firstname == value)
      return;

    firstname = value;
    OnPropertyChanged ("FirstName");
  }
}
```

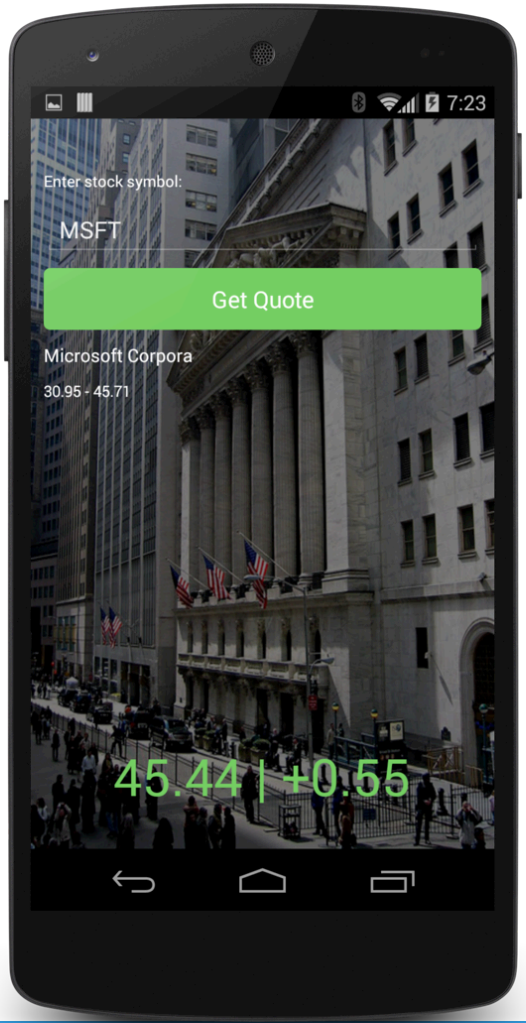
```
Label firstName = new Label ();
firstName.SetBinding (Label.TextProperty, "FirstName");
```

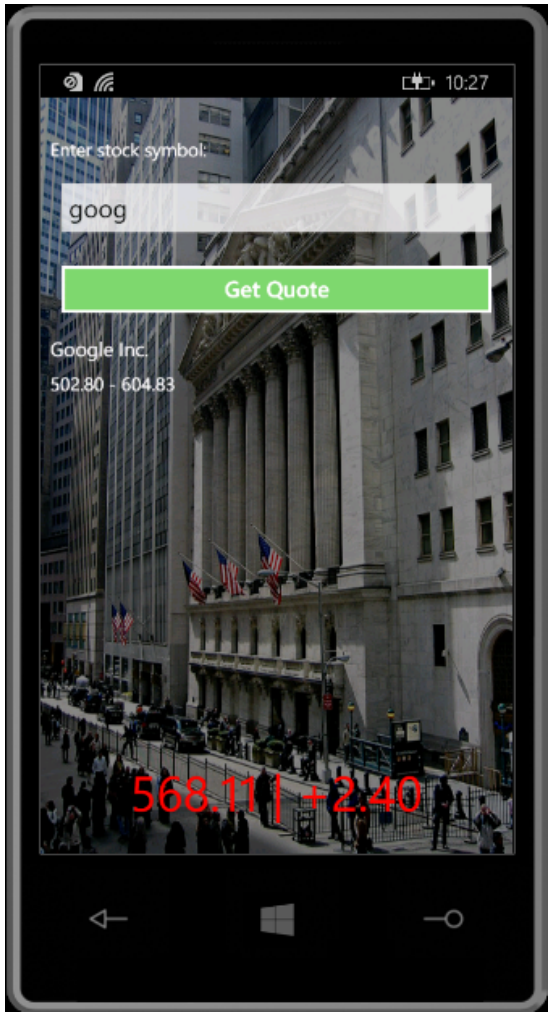
```
Entry firstEntry = new Entry ();
firstEntry.SetBinding<UserViewModel> (Entry.TextProperty, vm => vm.FirstName, BindingMode.TwoWay);
```

DEMO 2
Stocks App

www.github.com/JamesMontemagno/MyStocks.Forms

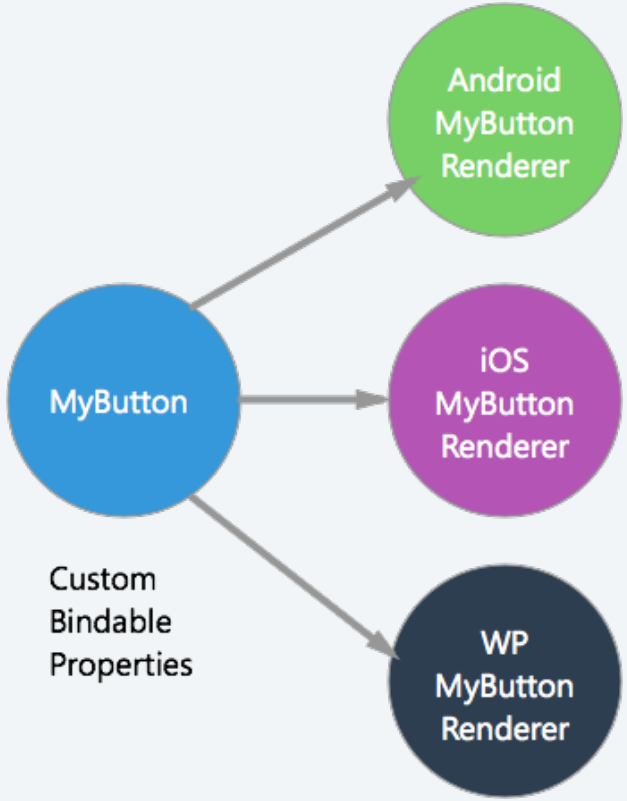

Carrier
7:28 PM
Enter stock symbol:
FB
Get Quote
Facebook, Inc.
41.44 - 76.74
74.92 | +0.965


7:23
Enter stock symbol:
MSFT
Get Quote
Microsoft Corpora
30.95 - 45.71
45.44 | +0.55


10:27
Enter stock symbol:
goog
Get Quote
Google Inc.
502.80 - 604.83
568.11 | +2.40

Extensibility

- Custom Renderer
  - Easily subclass any control you wish to customize.
  - Add your own custom controls.
    - Add custom bindable properties

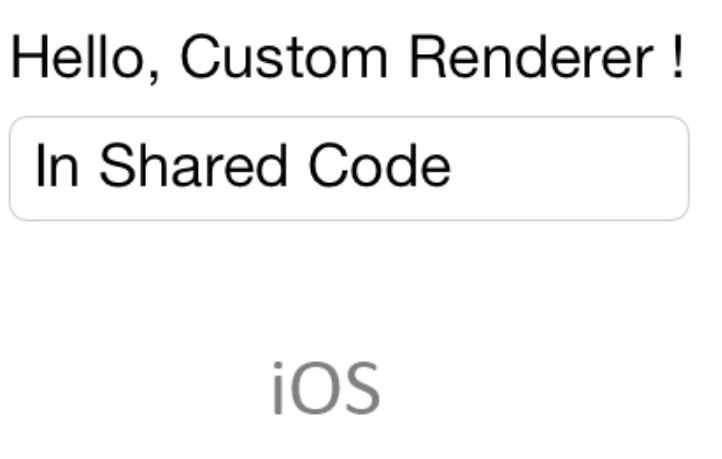


http://developer.xamarin.com/guides/cross-platform/xamarin-forms/custom-renderer/

Extensibility

- Standard Entry Control

```
new Entry {Text = "In Shared Code"}
```



Extensibility

- Step 1: Custom Xamarin.Forms Control

```
public class MyEntry : Entry {}
```

Extensibility

- Step 2: Use Custom Control in our shared Pages

```
public class MainPage : ContentPage
{
  public MainPage ()
  {
    Content = new StackLayout {
      Children = {
        new Label {
          Text = "Hello, Custom Renderer !",
        },
        new MyEntry {
          Text = "In Shared Code",
        }
      },
      VerticalOptions = LayoutOptions.CenterAndExpand,
      HorizontalOptions = LayoutOptions.CenterAndExpand,
    };
  }
}
```

Extensibility

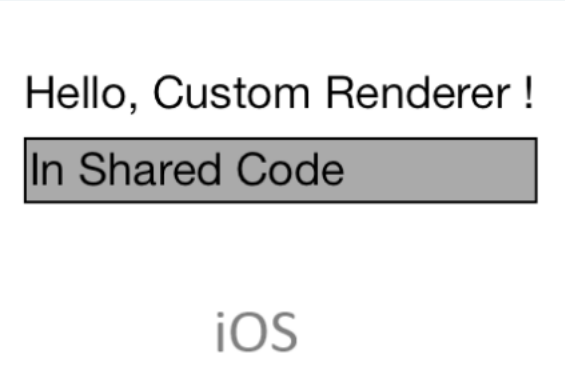
- Step 3: Implement Custom Renderer

```
public class MyEntryRenderer : EntryRenderer
{
  // Override the OnElementChanged method so we can tweak this renderer post-initial setup
  protected override void OnElementChanged (ElementChangedEventArgs<Entry> e)
  {
    base.OnElementChanged (e));
    if (e.OldElement == null) {   // perform initial setup
      // lets get a reference to the native control
      var nativeTextField = (UITextField) Control;
      // do whatever you want to the UITextField here!
      nativeTextField.BackgroundColor = UIColor.LightGray;
      nativeTextField.BorderStyle = UITextBorderStyle.Line;
    }
  }
}
```

Extensibility

- Step 4: Export Renderer

```
[assembly: ExportRenderer (typeof (MyEntry), typeof (MyEntryRenderer))]
```



Extensibility

- Dependency Service
  - Lets shared code access platform-specific SDK features via an Interface implementation

Extensibility

- Step 1: Interface

```
public interface ITextToSpeech
{
    void Speak (string text);
}
```

Extensibility

- Step 2: Implement on each Platform

```
public class TextToSpeech_iOS : ITextToSpeech
{
    public TextToSpeech_iOS () {}

    public void Speak (string text)
    {
        var speechSynthesizer = new AVSpeechSynthesizer ();

        var speechUtterance = new AVSpeechUtterance (text) {
            Rate = AVSpeechUtterance.MaximumSpeechRate/4,
            Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
            Volume = 0.5f,
            PitchMultiplier = 1.0f
        };

        speechSynthesizer.SpeakUtterance (speechUtterance);
    }
}
```

Extensibility

- Step 3: Register with Dependency Service

```
[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeech_iOS))]
```

Extensibility

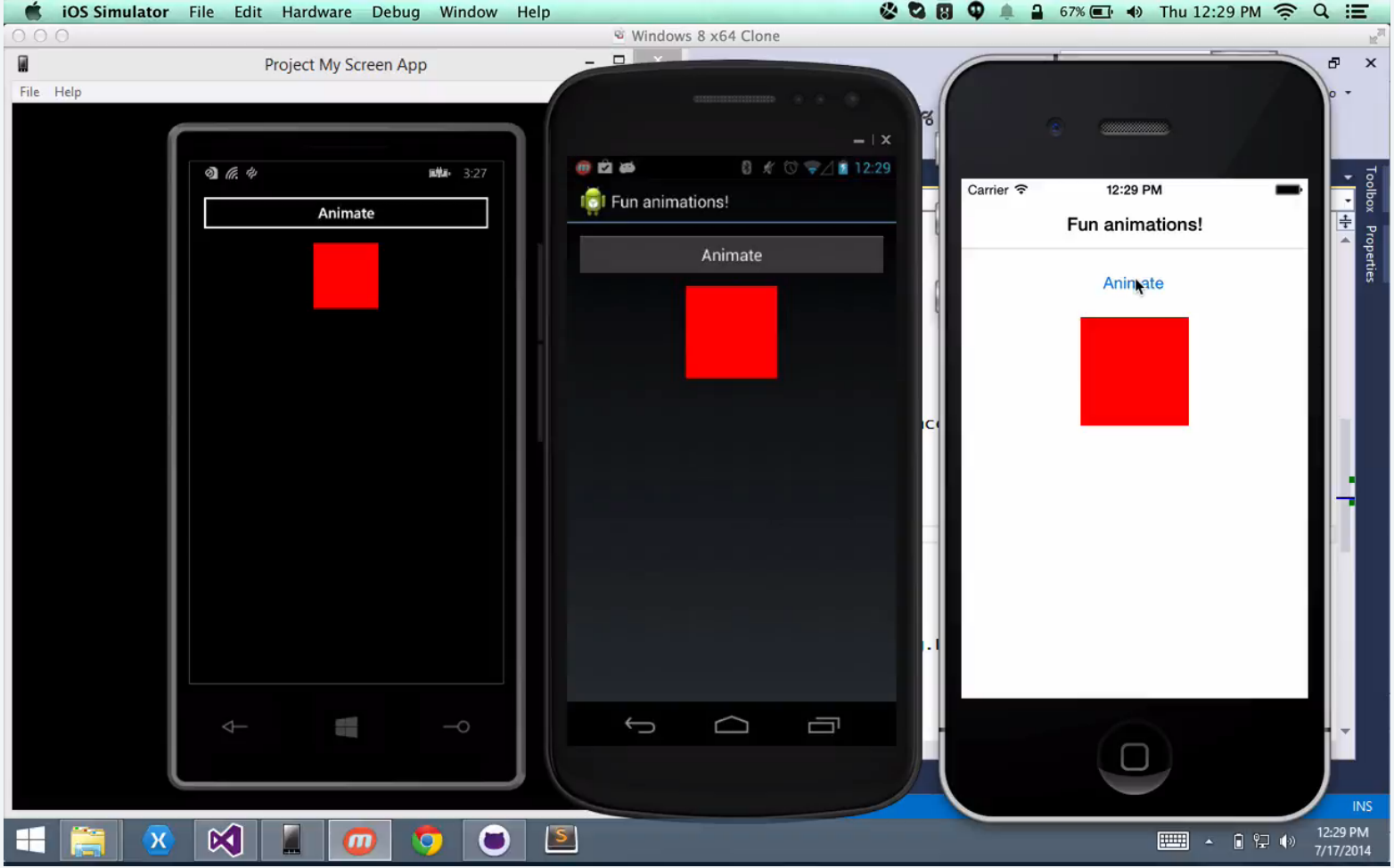
- Step 4: Access from shared code

```
speak.Clicked += (sender, e) => {
    DependencyService.Get<ITextToSpeech>().Speak("Hello from Xamarin Forms");
};
```

DEMO 3
Stocks App +
Text to Speech

www.github.com/JamesMontemagno/MyStocks.Forms

Animations API

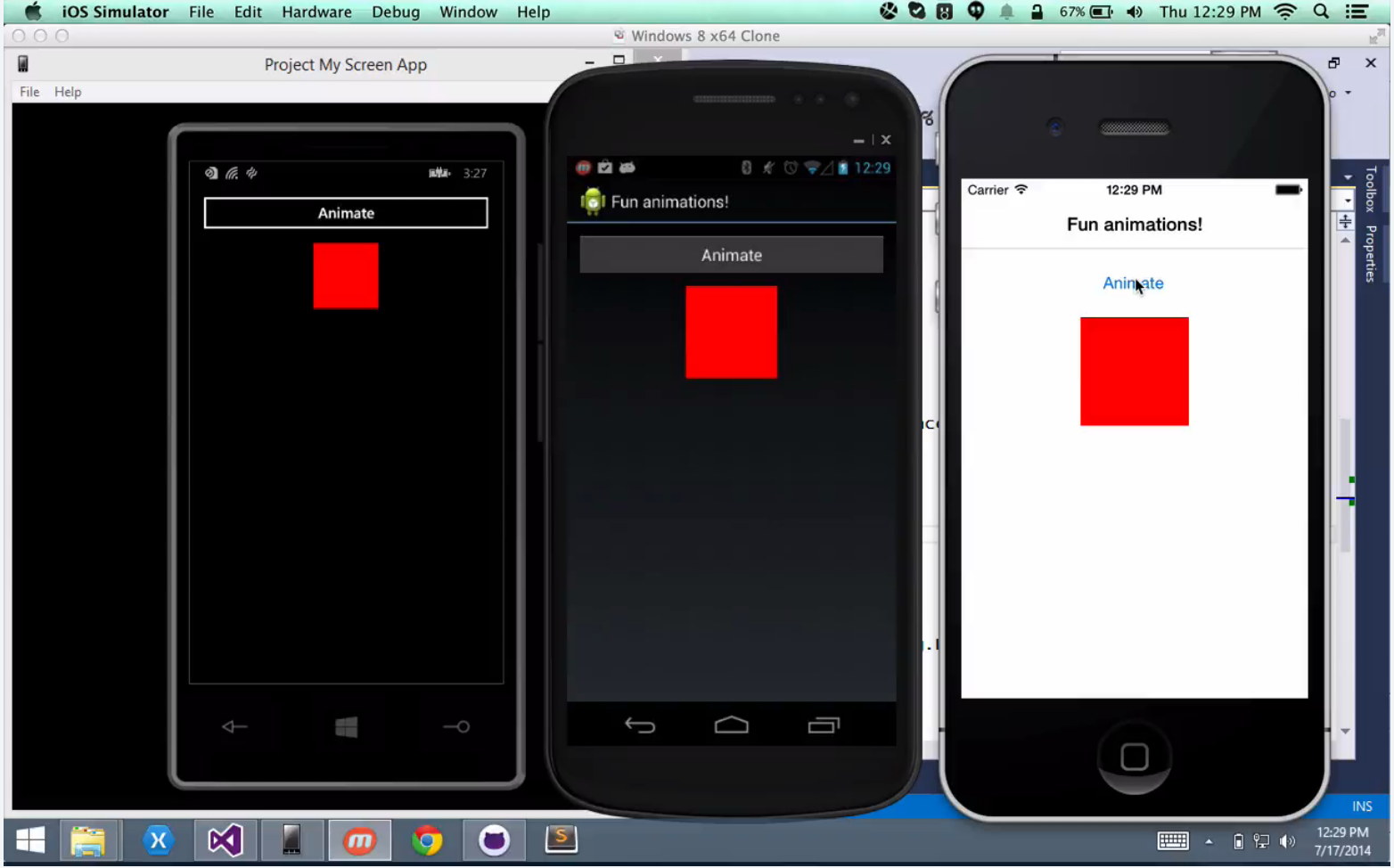


www.github.com/JamesMontemagno/FormsAnimations

```
button.Clicked += async (sender, args) =>
{
    button.IsEnabled = false;

    box.Color = Color.Green;

    var originalPosition = box.Bounds;
    var newPosition = box.Bounds;
    newPosition.Y = contentPage.Height - box.Height;

    await box.LayoutTo(newPosition, 2000, Easing.BounceOut);

    box.FadeTo(0, 2000);
    box.Color = Color.Yellow;

    await box.ScaleTo(2, 2000);

    box.FadeTo(1, 2000);
    await box.ScaleTo(1, 2000);

    box.Color = Color.Green;

    await box.LayoutTo(originalPosition, 2000, Easing.Linear);

    box.Color = Color.Red;
    button.IsEnabled = true;

};
```

```
button.Clicked += async (sender, args) =>
{
    button.IsEnabled = false;

    box.Color = Color.Green;

    var originalPosition = box.Bounds;
    var newPosition = box.Bounds;
    newPosition.Y = contentPage.Height - box.Height;

    await box.LayoutTo(newPosition, 2000, Easing.BounceOut);

    box.FadeTo(0, 2000);
    box.Color = Color.Yellow;

    await box.ScaleTo(2, 2000);

    box.FadeTo(1, 2000);
    await box.ScaleTo(1, 2000);

    box.Color = Color.Green;

    await box.LayoutTo(originalPosition, 2000, Easing.Linear);

    box.Color = Color.Red;
    button.IsEnabled = true;

};
```

```
button.Clicked += async (sender, args) =>
{
    button.IsEnabled = false;

    box.Color = Color.Green;

    var originalPosition = box.Bounds;
    var newPosition = box.Bounds;
    newPosition.Y = contentPage.Height - box.Height;

    await box.LayoutTo(newPosition, 2000, Easing.BounceOut);

    box.FadeTo(0, 2000);
    box.Color = Color.Yellow;

    await box.ScaleTo(2, 2000);

    box.FadeTo(1, 2000);
    await box.ScaleTo(1, 2000);

    box.Color = Color.Green;

    await box.LayoutTo(originalPosition, 2000, Easing.Linear);

    box.Color = Color.Red;
    button.IsEnabled = true;

};
```



www.github.com/JamesMontemagno/FormsAnimations

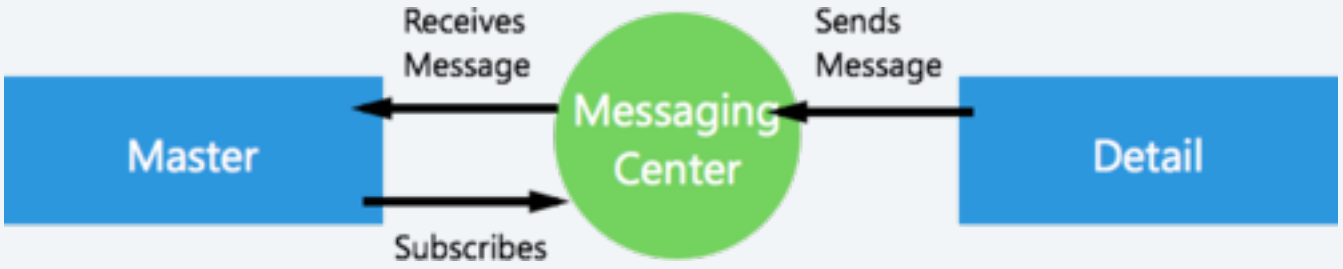
Navigation

- Root Page:
  - NavigationPage – Gives each page an INavigation

  - Standard Navigation
    - Navigation.PushAsync(page: nextPage);
    - Navigation.PopAsync();

  - Modal Navigation
    - Navigation.PushModalAsync(page: modalPage);
    - Navigation.PopModalAsync();

Messaging Center



- MessagingCenter.Subscribe<T>(object subscriber, string message, Action<T> callback);
- MessagingCenter.Send(T item, string message);

Messaging Center

- Master Page:

```
//Subscibe to insert expenses
MessagingCenter.Subscribe<TripExpense>(this, "AddNew", (expense) =>
{
  Expenses.Add(expense);
});
```

- Detail Page:

```
MessagingCenter.Send(expense, "AddNew");
```

Resources

- Documentation
  - http://developer.xamarin.com/guides/cross-platform/xamarin-forms/
- XAML Documentation
  - http://developer.xamarin.com/guides/cross-platform/xamarin-forms/xaml-for-xamarin-forms/
- Samples
  - https://github.com/xamarin/xamarin-forms-samples

Xamarin.com/University

Xamarin University

Unrivaled Mobile Development Training

Live unlimited mobile development training from mobile experts, in your time-zone, on your schedule and as often as you'd like.

Evolve 2014

Oct 6-10 | Atlanta | Still some tickets left!

Find Me:

@redth

http://redth.codes

Q & A

Get your free C# t-shirt:

xamarin.com/shirt