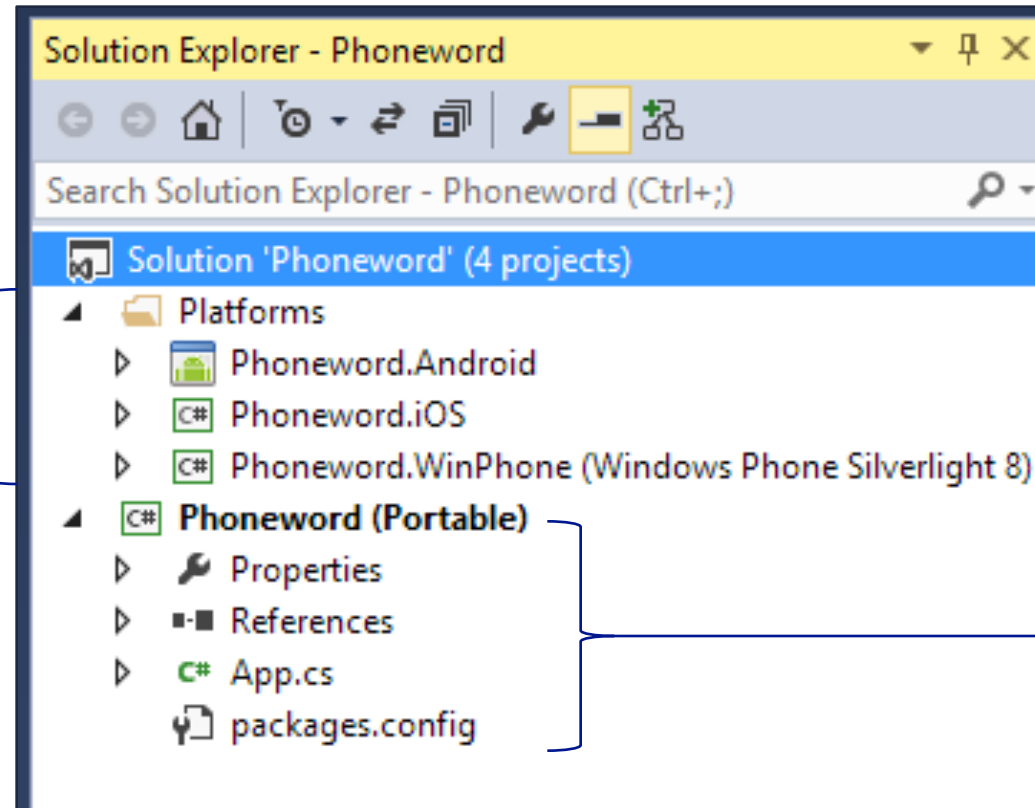


Xamarin.Forms: Our First App!

James Montemagno
@JamesMontemagno

Project Structure

Platform-specific projects act as "host" to create native application



Portable Class Library or Shared Project used to hold shared code that defines UI and logic

Xamarin.Forms Application

Application lifecycle:

- **OnStart**
- **OnSleep**
- **OnResume**

Properties dictionary (persists across app re-starts)

NOTE: beware of existing App class in Windows Phone projects.

```
public class App : Xamarin.Forms.Application
{
    public App ()
    {
        MainPage = new ContentPage { Title = "App Lifecycle Sample" }; // your page here
    }

    protected override void OnStart()
    {
        // Handle when your app starts
        Debug.WriteLine ("OnStart");
    }

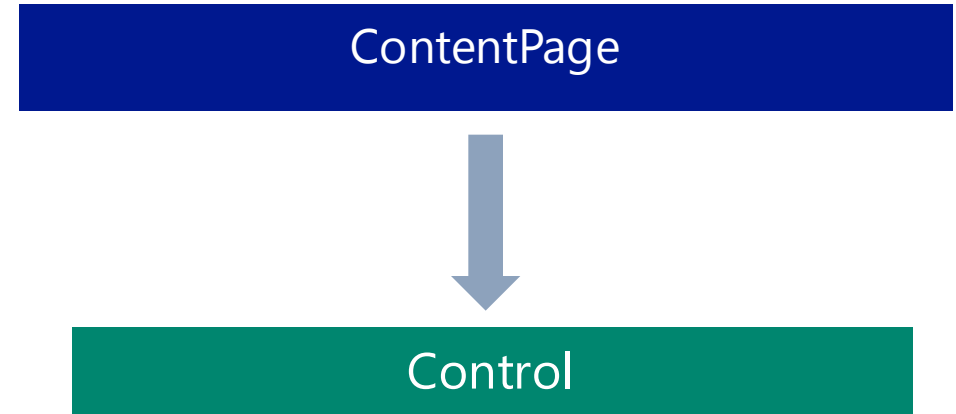
    protected override void OnSleep()
    {
        // Handle when your app sleeps
        Debug.WriteLine ("OnSleep");
    }

    protected override void OnResume()
    {
        // Handle when your app resumes
        Debug.WriteLine ("OnResume");
    }
}
```

```
Application.Current.Properties ["userInput"] = e.Text;
```

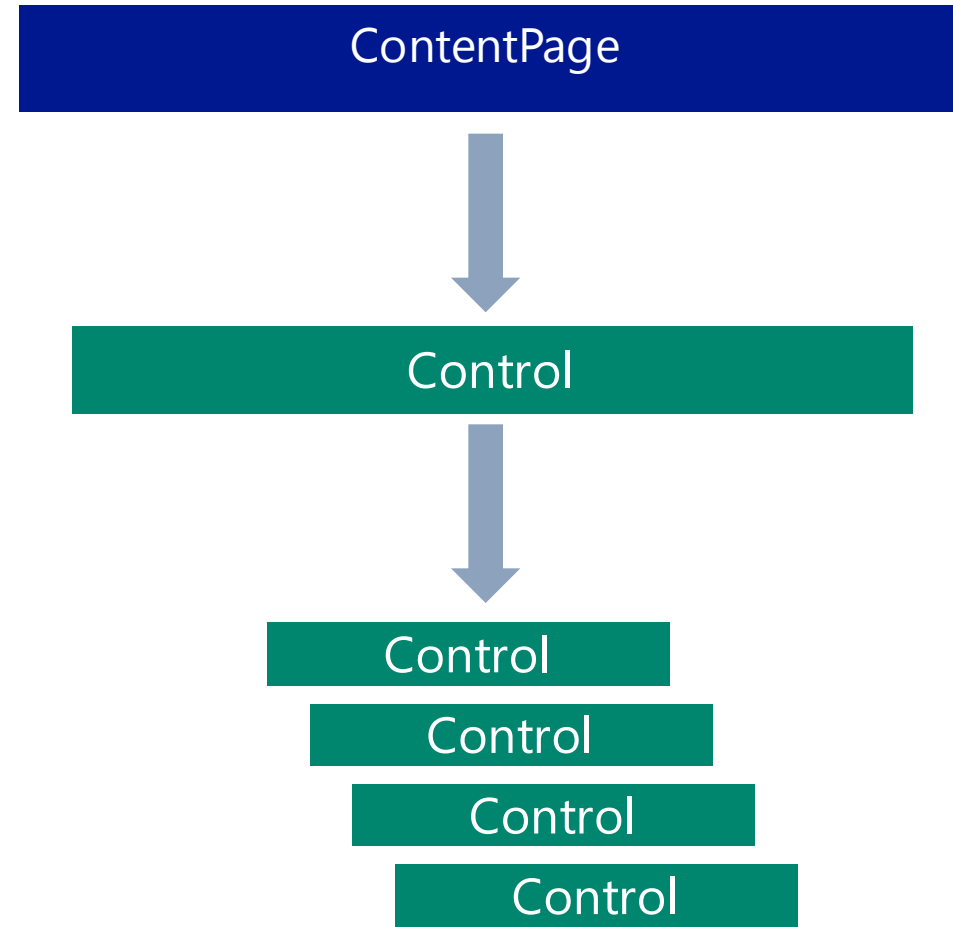
Pages

- Single screen of content
- ContentPage holds one visual element




Layout

- Layouts handle child elements
- Layouts come in two types: managed and unmanaged




Providing Behavior

Controls expose *properties* to alter visualization



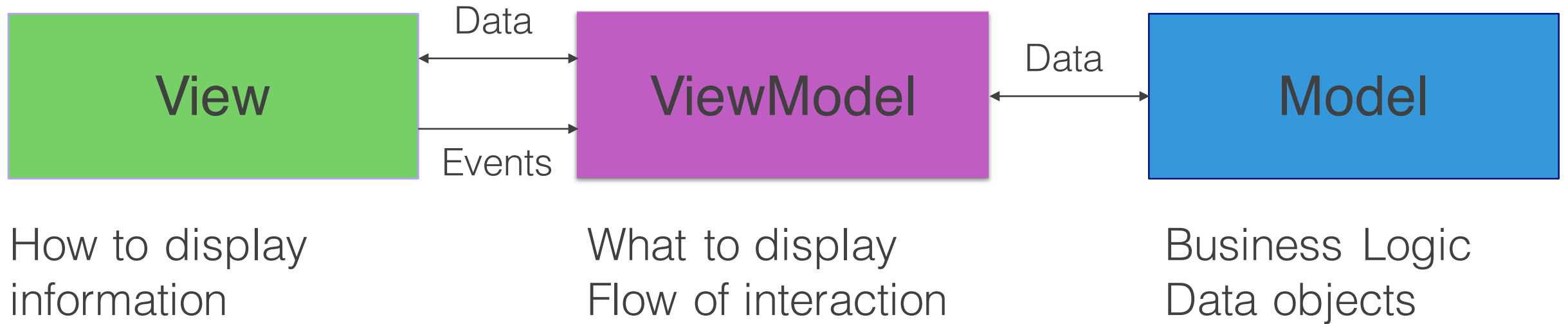
```
var entry = new Entry  
{  
    Placeholder = "Enter text",  
    Keyboard = Keyboard.Email  
};
```



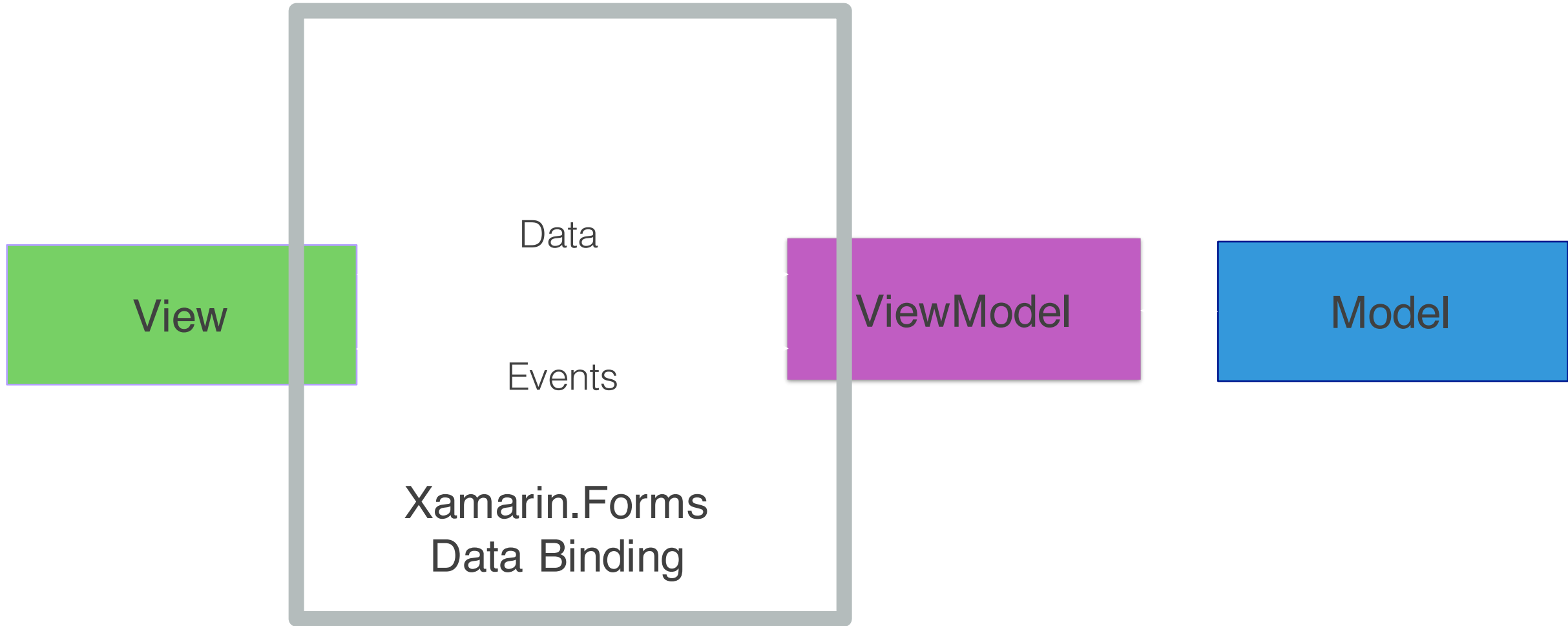
```
entry.TextChanged += (sender, e) => {  
    // Input changed  
};
```

Controls expose *events* to provide interactive behavior

Model-View-ViewModel



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);
```

Data Binding – XAML

```
<Label Text="{Binding FirstName}"/>
```

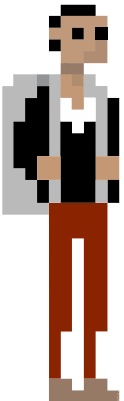
Data Binding – XAML

```
<Label Text="{Binding FirstName}"/>
```

```
<Entry Placeholder="FirstName"  
        Text="{Binding FirstName, Mode=TwoWay}"/>
```

Let's do it!

20 Minute Break



James
Montemagno
Developer Evangelist, Xamarin

james@xamarin.com

motzcod.es

[@JamesMontemagno](https://twitter.com/JamesMontemagno)