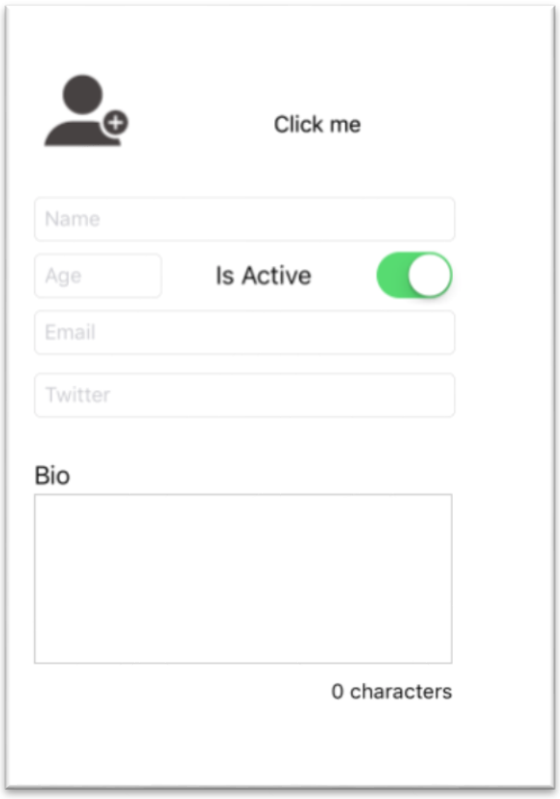
Xamarin.iOS – Creating a Simple View

In this exercise, you will create a layout for a simple user profile.

To learn more about the topics discussed in this workshop, refer to the Resources links available in the XamariniOSFundamentals_ViewsAndViewControllers.pdf in this directory and read through the online Developer documentation on the Apple and Xamarin sites.

The resulting user interface will look similar to the image below.



Open the solution in the Start folder

1. Open the MainViewController.storyboard
2. Ensure that your Toolbox is visible. If not, select View > Toolbox.
3. Drag the UI Controls to the form (to get it to represent the illustration above as closely as possible):
    a. Image View

In the Widget properties:

i. set Image to Images/ProfilePic.png
ii. Content Mode: Aspect Fit

b. Button (to the right of the Image View)

In the Widget properties, set Name to MyButton

c. TextField
    i. Set Placeholder to Name
d. TextField
    i. Set Placeholder to Age
    ii. Set Keyboard Type to Number Pad
e. TextField
    i. Set Placeholder to Email
    ii. Set Keyboard Type to Email address
f. TextField
    i. Set Placeholder to Twitter
    ii. Set Keyboard Type to Twitter
g. Label (Text = Is Active)
h. Switch (to the right of the Is Active label)
i. Label (Text = Bio)
j. Text View
    i. Clear the default text
    ii. Set Name to ContactBio
k. Label (Text = 0 characters)
    i. Set Name to CharsEnteredLabel

4. Next we want to set the button's title in the code behind, as well as the color of the font in the button's normal state, and selected state. To accomplish this, we need to use the button's SetTitle(string, UIControlState) method and SetTitleColor(UIColor, UIControlState) methods respectively. Do this within **ViewDidLoad**.

   Hint:

```
MyButton.SetTitle("Click me", UIControlState.Normal);
MyButton.SetTitleColor(UIColor.Black, UIControlState.Normal);
MyButton.SetTitleColor(UIColor.White, UIControlState.Selected);
```

5. Let's take a look at how we can register an eventhandler for the button when the user clicks on it. Remember, iOS exposes Touch events for user interaction. In this case, we will need to use TouchUpInside to react to a button click. Within **ViewDidLoad**, register an eventhandler for TouchUpInside, called **MyButtonTouchUpInside**. In this event handler we will change the BackgroundColor of the Button to a random color. Hint:

```
private void MyButtonOnTouchUpInside(object sender,
        EventArgs eventArgs)
{
    int r = random.Next(0, 255);
    int g = random.Next(0, 255);
    int b = random.Next(0, 255);

    MyButton.BackgroundColor = UIColor.FromRGB(r, g, b);
}
```

6. The next step is to add a border around the Contact Bio TextView. If you look at the Widget Properties, you'll notice that there is no option to set a border at this level. Instead, the border on the TextView is affected on the TextView's layer. In code, within **ViewDidLoad**, you can define the border color and border width on the TextView as follows:

```
ContactBio.Layer.BorderColor = new CGColor(0.5f, 0.5f);
ContactBio.Layer.BorderWidth = 1.0f;
```

7. To demonstrate how to subscribe to event handler when change is affected on the Text Field, register an event handler to the ContactBio's **Changed** event. In the Changed event, we will set the **CharsEnteredLabel** text to display the number of characters entered in the **ContactBio** field. Hint:

```
private void ContactBioOnChanged(object sender, EventArgs eventArgs)
{
   this.CharsEnteredLabel.Text = $"{ContactBio.Text.Length} characters";
}
```

8. Override the **ViewDidUnload** event, and unregister the ContactBio.Changed event handler:

```
public override void ViewDidUnload()
{
    base.ViewDidUnload();
    this.ContactBio.Changed -= ContactBioOnChanged;
}
```

9. Build and run the application. If you deploy to the simulator, be sure that the soft input keyboard is displayed (to simulate entry from a device).
10. Click on the Button multiple times.
11. Click on each Text Field (name, email, age, twitter). What do you notice about the iOS keyboard when focus is on each of the fields?
12. Enter text in the Bio text view. What issue do you notice?

One issue you may have noticed when entering text in the Bio field, is that the keyboard covers the field, and you cannot see the label below the Bio field. One way to solve this problem, is to programmatically resize the frame and scroll the view. For details on this, and to implement this type of behaviour, refer to this blog post: http://www.gooorack.com/2013/08/28/xamarin-moving-the-view-on-keyboard-show/

Alternatively, copy and paste the following into the MainViewController class:

```
#region KeyBoard Notification
        //Credits: http://www.gooorack.com/2013/08/28/xamarin-moving-the-view-on-keyboard-show/
```

```
private UIView activeView;
private nfloat scrollAmount = 0.0f;
private nfloat bottom = 0.0f;
private nfloat offset = 30.0f;
private bool moveViewUp;

private void KeyBoardUpNotification(NSNotification notification)
{
    // get the keyboard size
    CGRect r = UIKeyboard.BoundsFromNotification(notification);

    // Find what opened the keyboard
    foreach (UIView view in View.Subviews)
    {
        if (view.IsFirstResponder)
        {
            activeView = view;
            break;
        }

    }

    // Bottom of the controller = initial position + height + offset
    bottom = (activeView.Frame.Y + activeView.Frame.Height + offset);

    // Calculate how far we need to scroll
    scrollAmount = (r.Height - (View.Frame.Size.Height - bottom));

    // Perform the scrolling
    if (scrollAmount > 0)
    {
        moveViewUp = true;
        ScrollTheView(moveViewUp);
    }
    else
    {
        moveViewUp = false;
        ResetView();
    }

}

private void KeyBoardDownNotification(NSNotification notification)
{
    if (moveViewUp)
    {
        ScrollTheView(false);
    }
}
```

```
        private void ResetView()
        {
            UIView.BeginAnimations(string.Empty, System.IntPtr.Zero);
            UIView.SetAnimationDuration(0.3);
            View.Frame = UIScreen.MainScreen.Bounds;
            UIView.CommitAnimations();
        }

        private void ScrollTheView(bool move)
        {

            // scroll the view up or down
            UIView.BeginAnimations(string.Empty, System.IntPtr.Zero);
            UIView.SetAnimationDuration(0.3);

            CGRect frame = View.Frame;

            if (move)
            {
                frame.Y -= scrollAmount;
            }
            else
            {
                frame.Y += scrollAmount;
                scrollAmount = 0;
            }

            View.Frame = frame;
            UIView.CommitAnimations();
        }
        #endregion
```

13. In the ViewDidLoad, register observers with NSNotificationCenter to handle when the keyboard is displayed and hidden. Hint:

```
// Keyboard popup
NSNotificationCenter.DefaultCenter.AddObserver
      (UIKeyboard.DidShowNotification, KeyBoardUpNotification);

// Keyboard Down
NSNotificationCenter.DefaultCenter.AddObserver
            (UIKeyboard.WillHideNotification, KeyBoardDownNotification);
```

14. Build and run the application. Set focus to the Bio field. What do you notice now?

15. Set focus to another text field on the form. Does the form view reset back to its natural state?

BONUS exercise:

Delete all of the UI Controls from the storyboard.

Create them programmatically within **ViewDidLoad** and add them to the ViewController's main view**.**