**Hello, iOS**

Introduction to iOS Development with Xamarin

**Hello.iOS Quickstart**

This guide describes how to create an application that translates an alphanumeric phone number entered by the user into a numeric phone number, and then calls that number. The final application looks like this:

**Requirements**

iOS development with Xamarin requires:

* A Mac running OS X El Capitan (10.11) or above.
* Latest version of Xcode and iOS SDK installed from the [App Store](https://itunes.apple.com/us/app/xcode/id497799835?mt=12) .

Xamarin.iOS works with any of the following setups:

* Latest version of Visual Studio for Mac on a Mac that fits the above specifications.
* Latest version of Visual Studio 2015 or 2017 Professional or higher on Windows 7 or above, paired with a Mac build host that fits the above specifications.

The [Xamarin.iOS Windows Installation guide](https://developer.xamarin.com/guides/ios/getting_started/installation/windows/) is available for step-by-step installation instructions.

Before getting started, download the [Xamarin App Icons set](https://developer.xamarin.com/guides/ios/getting_started/hello,_iOS/hello,iOS_quickstart/Resources/XamarinAppIconsandLaunchImages.zip).

**Visual Studio Walkthrough**

These instructions describe how to create an application called Phoneword that translates an alphanumeric phone number into a numeric phone number.

1. Launch Visual Studio from the **Start** menu:

In the search box under **New Solution** enter *Single View App*, and select **Single View App (iPhone)** to create a new Xamarin.iOS solution:

1. Name the Project and Solution Phoneword, as illustrated below:
2. Press **OK** to create the new Project
3. Confirm that the Xamarin Mac Agent icon in the toolbar is green.

If it isn't connected, follow the steps in the [configuration guide](https://developer.xamarin.com/guides/ios/getting_started/installation/windows/connecting-to-mac/).

1. Open the **Main.storyboard** file in the iOS Designer by double-clicking on it in the **Solution Explorer**:
2. In the Designer Toolbar, click on **View As** and change it from *Generic* to *iPhone 6*:
3. Open the **Toolbox** tab, type “label” into the search bar and drag a **Label** onto the design surface (the area in the center):
4. Next, grab the handles of the *Dragging Controls* (the circles around the control – make sure they are circles they *don't* look like 'bones') and make the label wider:
5. With the **Label** selected on the design surface, use the **Properties Windows** to change the **Text** property of the **Label** to "Enter a Phoneword:"

**Note:** You can bring up the **Properties** or **Toolbox** at any time by navigating to the **View** menu.

1. Search for “text field” inside the Toolbox and drag a **Text Field** from the **Toolbox** onto the design surface and place it under the **Label**. Adjust the width until the **Text Field** is the same width as the **Label**:
2. With the **Text Field** selected on the design surface, change the **Text Field**’s **Name** property in the Identity section of the **Properties** to PhoneNumberText, and change the **Text** property to "1-855-XAMARIN":
3. Drag a **Button** from the **Toolbox** onto the design surface and place it under the **Text Field**. Adjust the width so the **Button** is as wide as the **Text Field** and **Label**:
4. With the **Button** selected on the design surface, change the **Name** property in the **Identity** section of the **Properties** to TranslateButton. Change the **Title** property to "Translate":
5. Repeat the previous two steps and drag a **Button** from the **Toolbox** onto the design surface and place it under the first **Button**. Adjust the width so the **Button** is as wide as the first **Button**:
6. With the second **Button** selected on the design surface, change the **Name** property in the **Identity** section of the **Properties** to CallButton. Change the **Title** property to "Call":

Save the changes by navigating to **File > Save All** or by pressing **Ctrl + s**.

1. Code to translate phone numbers from alphanumeric to numeric. To do this, first add a new file to the Project by right-clicking on the **Phoneword** Project in the **Solution Explorer** and choosing **Add > New Item...** or pressing **Ctrl + Shift + A**:
2. In the **New File** dialog, select **Apple > Class** and name the new file PhoneTranslator:

⚠️

Make sure that you select the 'class' template that has a C# in the icon. Otherwise you may not be able to reference this new class.

1. This creates a new C# class. Remove all the template code and replace it with the following code:
2. using System.Text;
3. using System;
4. namespace Phoneword
5. {
6. public static class PhoneTranslator
7. {
8. public static string ToNumber(string raw)
9. {
10. if (string.IsNullOrWhiteSpace(raw)) {
11. return "";
12. } else {
13. raw = raw.ToUpperInvariant();
14. }
15. var newNumber = new StringBuilder();
16. foreach (var c in raw)
17. {
18. if (" -0123456789".Contains(c)) {
19. newNumber.Append(c);
20. } else {
21. var result = TranslateToNumber(c);
22. if (result != null) {
23. newNumber.Append(result);
24. }
25. }
26. // otherwise we've skipped a non-numeric char
27. }
28. return newNumber.ToString();
29. }
30. static bool Contains (this string keyString, char c)
31. {
32. return keyString.IndexOf(c) >= 0;
33. }
34. static int? TranslateToNumber(char c)
35. {
36. if ("ABC".Contains(c)) {
37. return 2;
38. } else if ("DEF".Contains(c)) {
39. return 3;
40. } else if ("GHI".Contains(c)) {
41. return 4;
42. } else if ("JKL".Contains(c)) {
43. return 5;
44. } else if ("MNO".Contains(c)) {
45. return 6;
46. } else if ("PQRS".Contains(c)) {
47. return 7;
48. } else if ("TUV".Contains(c)) {
49. return 8;
50. } else if ("WXYZ".Contains(c)) {
51. return 9;
52. }
53. return null;
54. }
55. }

}

Save the **PhoneTranslator.cs** file and close it.

1. Double-click on **ViewController.cs** in the **Solution Explorer** to open it, so that logic can be added to handles interactions with the buttons:
2. Begin by wiring up the TranslateButton. In the **ViewController** class, find the ViewDidLoad method. Add the following button code inside ViewDidLoad, beneath the base.ViewDidLoad() call:
3. string translatedNumber = "";
4. TranslateButton.TouchUpInside += (object sender, EventArgs e) => {
5. // Convert the phone number with text to a number
6. // using PhoneTranslator.cs
7. translatedNumber = PhoneTranslator.ToNumber(PhoneNumberText.Text);
8. // Dismiss the keyboard if text field was tapped
9. PhoneNumberText.ResignFirstResponder ();
10. if (translatedNumber == "") {
11. CallButton.SetTitle ("Call", UIControlState.Normal);
12. CallButton.Enabled = false;
13. }
14. else {
15. CallButton.SetTitle ("Call " + translatedNumber, UIControlState.Normal);
16. CallButton.Enabled = true;
17. }

};

Include using Phoneword\_iOS; if the file's namespace is different.

1. Add code to respond to the user pressing the second button, which is named CallButton. Place the following code below the code for the TranslateButton and add using Foundation; to the top of the file:
2. CallButton.TouchUpInside += (object sender, EventArgs e) => {
3. var url = new NSUrl ("tel:" + translatedNumber);
4. // Use URL handler with tel: prefix to invoke Apple's Phone app,
5. // otherwise show an alert dialog
6. if (!UIApplication.SharedApplication.OpenUrl (url)) {
7. var alert = UIAlertController.Create ("Not supported", "Scheme 'tel:' is not supported on this device", UIAlertControllerStyle.Alert);
8. alert.AddAction (UIAlertAction.Create ("Ok", UIAlertActionStyle.Default, null));
9. PresentViewController (alert, true, null);
10. }

};

1. Save the changes, and then build the application by choosing **Build > Build Solution** or pressing **Ctrl + Shift + B**. If the application compiles, a success message will appear at the bottom of the IDE:

If there are errors, go through the previous steps and correct any mistakes until the application builds successfully.

1. The basic application functionality is now working and it’s time to add the finishing touches. Edit the application name and icons in the Project Properties To do this right-click on the Project in the **Solution Explorer** and select properties. Go to the iOS Application Tab.:
2. In the **iOS Application Target** section, change the **Application Name** to "Phoneword":
3. To set application icon, an Asset Catalog that will hold all the images is required. To create this, right-click on **Asset Catalogs** in the **Solution Explorer** and select **Add Asset Catalog**:

Name the Asset Catalog **Media** and press **Add**:

1. To set application icons and launch images, first download the [Xamarin App Icons set](https://developer.xamarin.com/guides/ios/getting_started/hello,_iOS/hello,iOS_quickstart/Resources/XamarinAppIconsandLaunchImages.zip). To start adding the icons, click directly on the (58x58) icon placeholder and browse to your folder location. Select the matching icon from the Xamarin App Icons directory:

Continue filling in all the icons. Visual Studio will replace the placeholders with the app icons:

1. Go to the application's Project options to tell the application to use the Asset Catalog for app icons. To do this right-click on the Project name and browse to **Properties**. Locate the **iOS Application** section. Under **App, Spotlight and Settings**, set the Asset Catalogs dropdown to **Media > App Icons**:
2. Finally, test the application in the **iOS Simulator**. In the IDE toolbar, choose **Debug** and **iPhone 6 iOS x.x** from the drop down menus, and press **Start** (the green triangle that resembles a Play button):

**Note:** At present, due to a requirement from Apple, it may be necessary to have a a development certificate or *signing identity* in order to build you code for device or simulator. Follow the steps in the [Device Provisioning guide](https://developer.xamarin.com/guides/ios/getting_started/installation/device_provisioning/#Requesting_a_Development_Certificate) to set this up.

1. This will launch the application inside the iOS Simulator:

Phone calls are not supported in the iOS Simulator; instead, an alert dialog will display when trying to place a call:

Congratulations on completing your first Xamarin.iOS application!