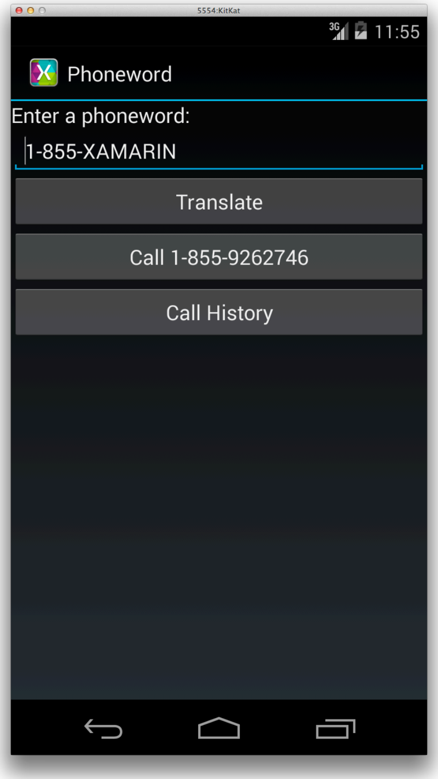
Introduction to Android with Xamarin Studio

Lab 2

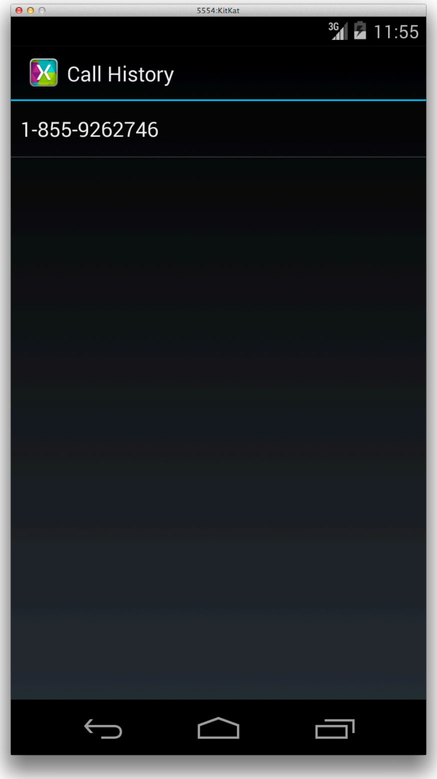
Xamarin Inc.

Lab Goals

In this lab, we'll explore how to create multi-screened Android applications using Activities and Intents. We'll do this by adding support for navigation between multiple screens by extending the **PhonewordAndroid** application we created in Lab 1 to include a second screen that contains the call history, as illustrated below:



1-855-XAMARIN



1-855-9262746

Lab 2

In this lab, we'll be extending our **PhonewordAndroid** application from Lab 1 with a new button that will navigate to a second Activity to display a **Call History** screen.

Open our solution in Xamarin Studio

1. Open your virtual environment or log onto your physical development environment with the Android development tools. Please contact your instructor, if there are any issues.
2. Navigate to the following folder location in the lab environment:

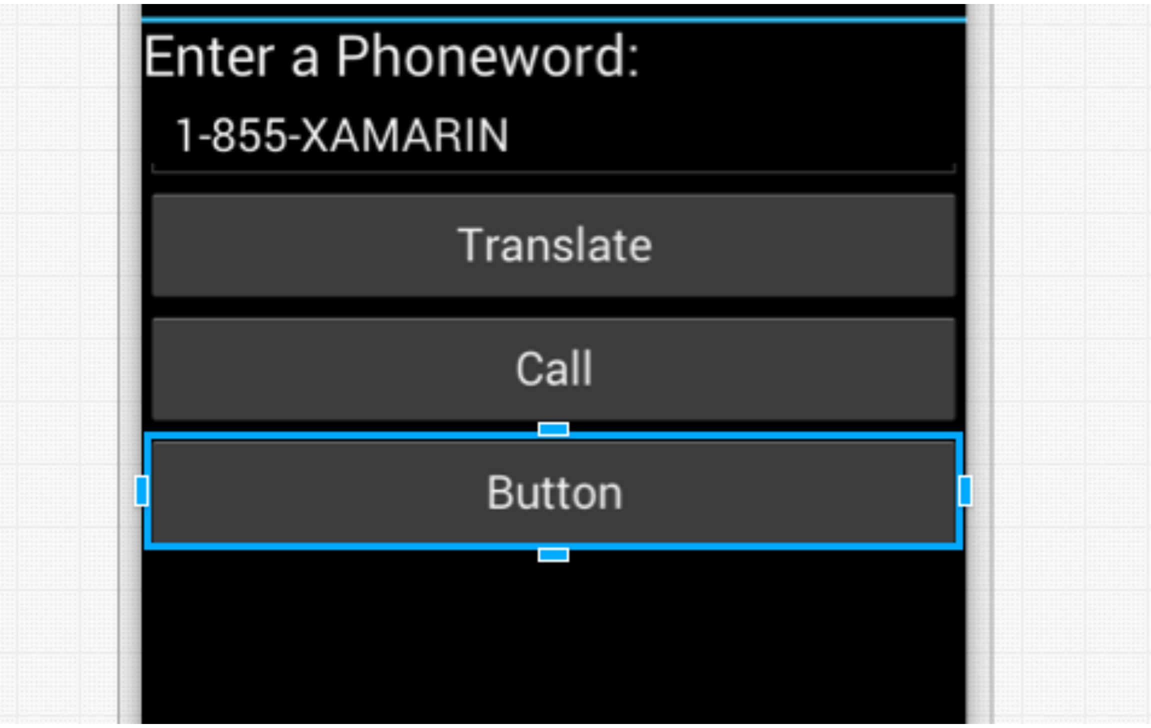
```
Xamarin University/Intro to Android/Lab 02
Resources/PhonewordAndroid_Start
```

3. Double-click the **Phoneword.sln** file. This solution is similar to the **PhonewordAndroid** application we completed in Lab 1. The changes made to facilitate this lab will be pointed out as necessary.

Adding the Call History Button

We are going to add a **Call History** button to the main screen of our **PhonewordAndroid** app to display any calls the user has initiated from within our app.

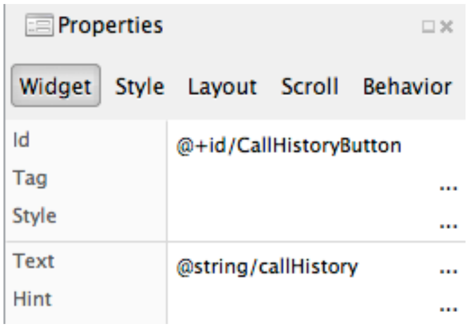
1. Open the **Main.axml** layout file in the **Resources/layout** folder, just expand the folders and then double-click on the file to open it in the designer.
2. Drag a new button onto the design surface directly under the existing **Call** button, it should look something like:



3. Select the **Widget** tab on the **Properties Pad** and make the following changes to the new Button:

Property	Value
Id	@+id/CallHistoryButton
Text	@string/callHistory

It will look something like:



Id	@+id/CallHistoryButton
Tag	...
Style	...
Text	@string/callHistory ...
Hint	...

4. Notice that the Button displays exactly what you placed into the Text property – we want this to come from our string resources so open the **Strings.xml** file in the **Resources/values** folder.

5. Add a new string into the file with the name "`callHistory`" and the value "`Call History`", it should look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Click Me!</string>
    <string name="app_name">PhonewordAndroid</string>
    <string name="callHistory">Call History</string>
</resources>
```

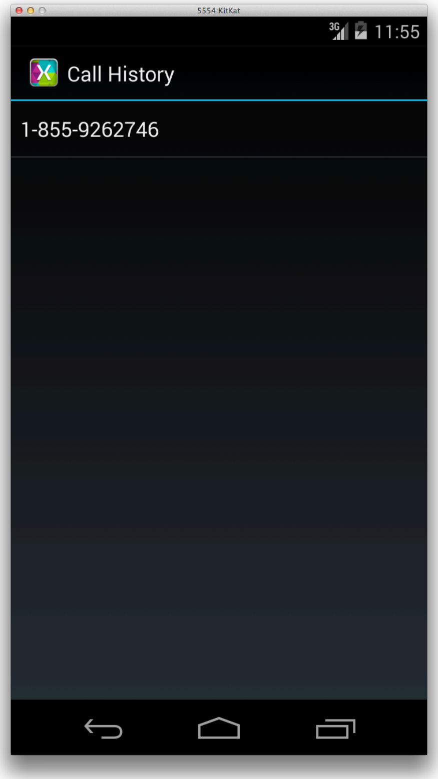
6. Switch back to the **Main.axml** file and verify that the button now displays "Call History".

7. On the **Behavior** tab of the **Properties Pad**, change the enabled state of the Button to be false. The Button in the designer should transition to a disabled state.

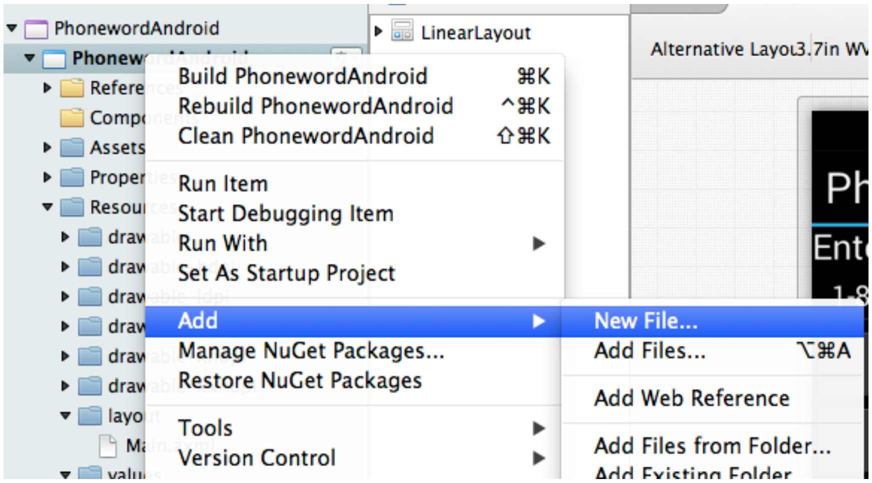
8. **File > Save All** (or **Shift+Command+S**).

Adding the Call History Activity

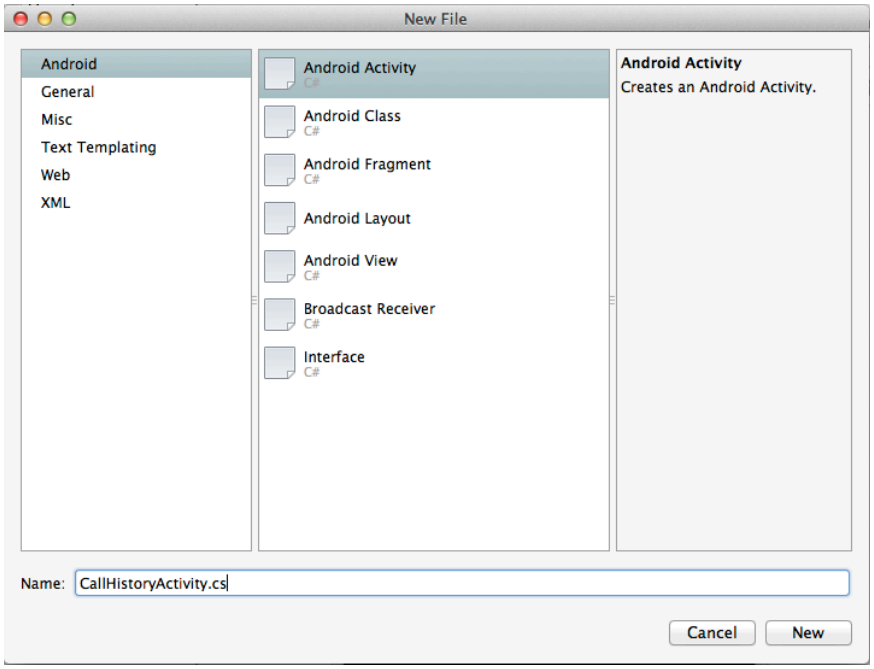
We will display a list of call placed within our application through a **ListActivity** titled **Call History** which will be shown when the user clicks on the Call History Button we just added. The **ListActivity** will show call activity as shown in the screen shot below:



1. Right-click on the **PhonewordAndroid** project and select **Add > New File**:



2. From the New File dialog, select the **Android Activity** item and set the **Name** to **CallHistoryActivity.cs**. Click **New** to create the Activity.



Name: CallHistoryActivity.cs

3. Open the **CallHistoryActivity.cs** file, it currently derives from the base **Activity** type, so change this to be **ListActivity** that displays lists of items.

```
[Activity(Label = "CallHistoryActivity")]
public class CallHistoryActivity : ListActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);

        // Create your application here
    }
}
```

4. Next, change the **Label** property on the **ActivityAttribute** to be "@string/callHistory" so we get the proper title bar caption.

```
[Activity(Label = "@string/callHistory")]
public class CallHistoryActivity : ListActivity
{
```

5. Next, modify the **OnCreate** method with the following code to get a string array passed from the caller and display it as the data for the **ListActivity**.

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);

    var phoneNumbers = Intent.Extras.GetStringArrayList("phone_numbers")
                       ?? new string[0];
    this.ListAdapter = new ArrayAdapter<string>(this,
                      Android.Resource.Layout.SimpleListItem1, phoneNumbers);
}
```

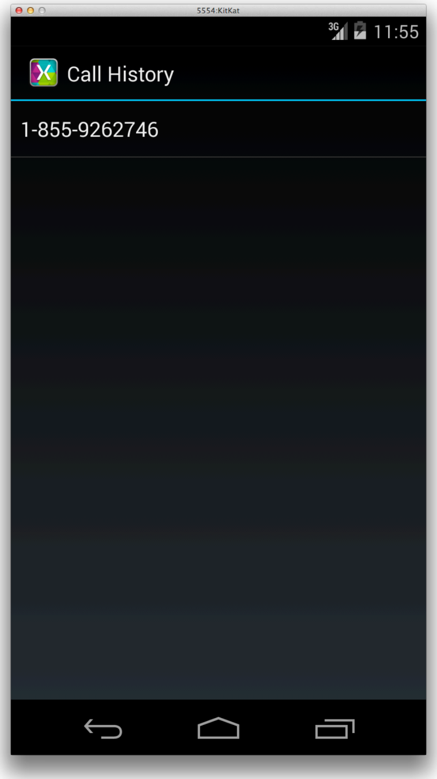
6. Open the **MainActivity.cs** source file and locate the **OnCreate** method – at the bottom you will find a **TODO** for this step with some commented-out code that handles the **CallHistoryButton Click** event and displays the **CallHistoryActivity** and passes a collection of strings to display. Go ahead and uncomment that code.

```
// TODO: Step 6 - handle the Call History button
Button CallHistoryButton = FindViewById<Button>(Resource.Id.CallHistoryButton);
CallHistoryButton.Click += (sender, e) =>
{
   var intent = new Intent(this, typeof(CallHistoryActivity));
   intent.PutStringArrayListExtra("phone_numbers", _phoneNumbers);
   StartActivity(intent);
};
```

7. Next, we need a collection to hold the call history, and some code to add each dialed number to that collection.
   a. At the top of the class you will find a `List<string>` named `_phoneNumbers` which is commented out – uncomment this.
   b. In the **callButton Click** handler, uncomment the code to add the translated number to the call history collection and to enable the **CallHistory Button**.

Testing our Application

1. Click the **Play** button to build and run our app.
2. Click the **Translate** button, followed by the **Call 1-855-9262746** button.
3. Click the **Call** button to dial the call and then end the simulated call.
4. Click the **Call History** button. We see that navigating to the call history after dialing shows the list of numbers dialed:



Call History

1-855-9262746

Summary

Congratulations! In this lab, we introduced how to create multi-screened Android applications and how to pass data from one screen to another.