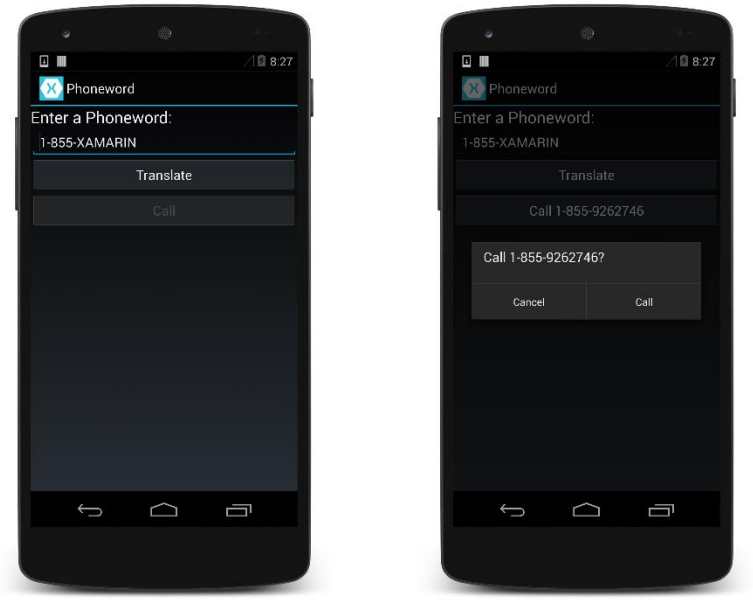
Hello, Android Multiscreen

Handling Navigation with Xamarin.Android

In this two-part guide, we expand the Phoneword application we created in the Hello, Android guide to handle a second screen. Along the way we'll introduce the basic Android Application Building Blocks and dive deeper into Android architecture as we develop a better understanding of Android application structure and functionality.

Hello, Android Multiscreen Quickstart

In the walkthrough part of this guide we'll add a second screen to our Phoneword application to keep track of the history of numbers called using our app. The final application will have a second screen that displays the call history, as illustrated by the following screenshots:



In the accompanying Deep Dive, we'll review what we've built and discuss architecture, navigation, and other new Android concepts that we encounter along the way.

Let's get started!

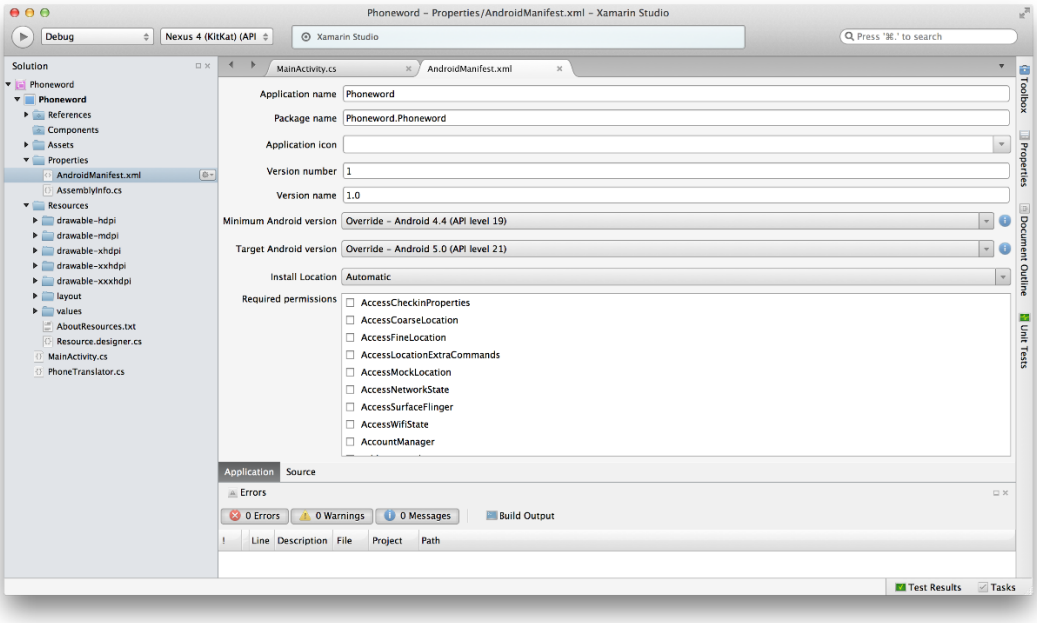
Requirements

Because this guide picks up where Hello, Android leaves off, it requires completion of the Hello, Android Quickstart. If you would like to jump directly into the walkthrough below, you can download the completed version of Phoneword (from the Hello, Android Quickstart) and use it to start the walkthrough.

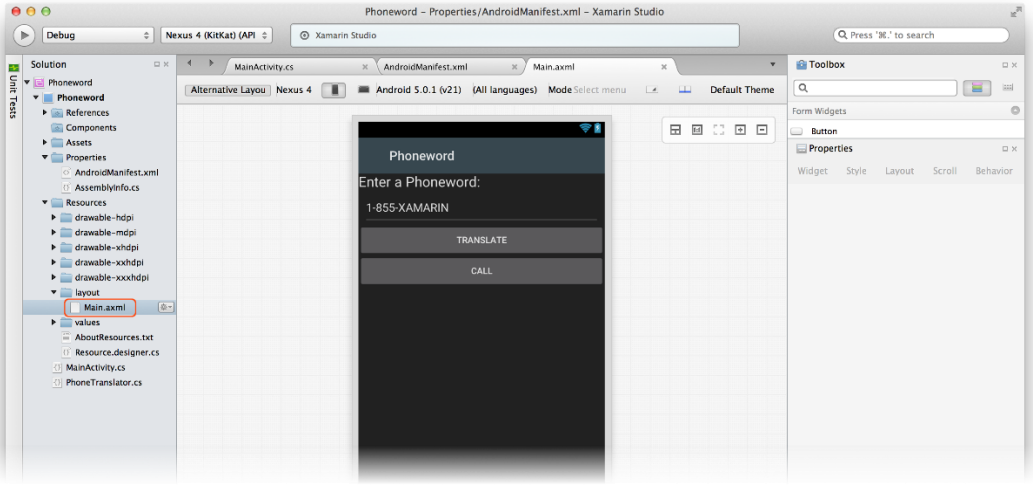
Walkthrough

In this walkthrough we'll add a **Call History** screen to our *Phoneword* application.

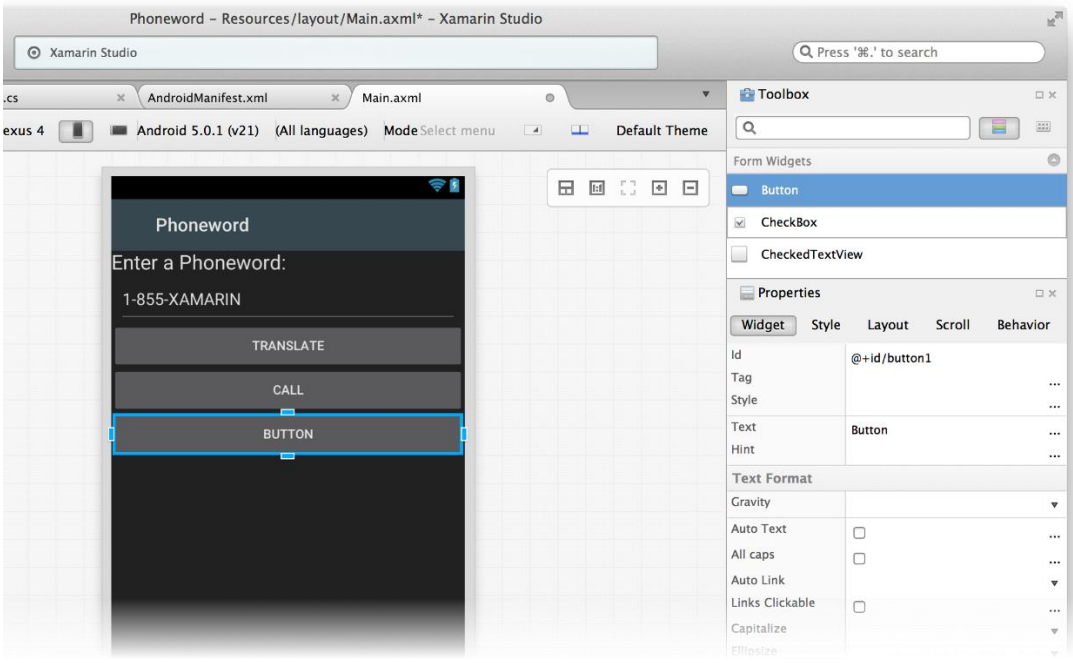
1. Let's open the **Phoneword** project in Xamarin Studio:



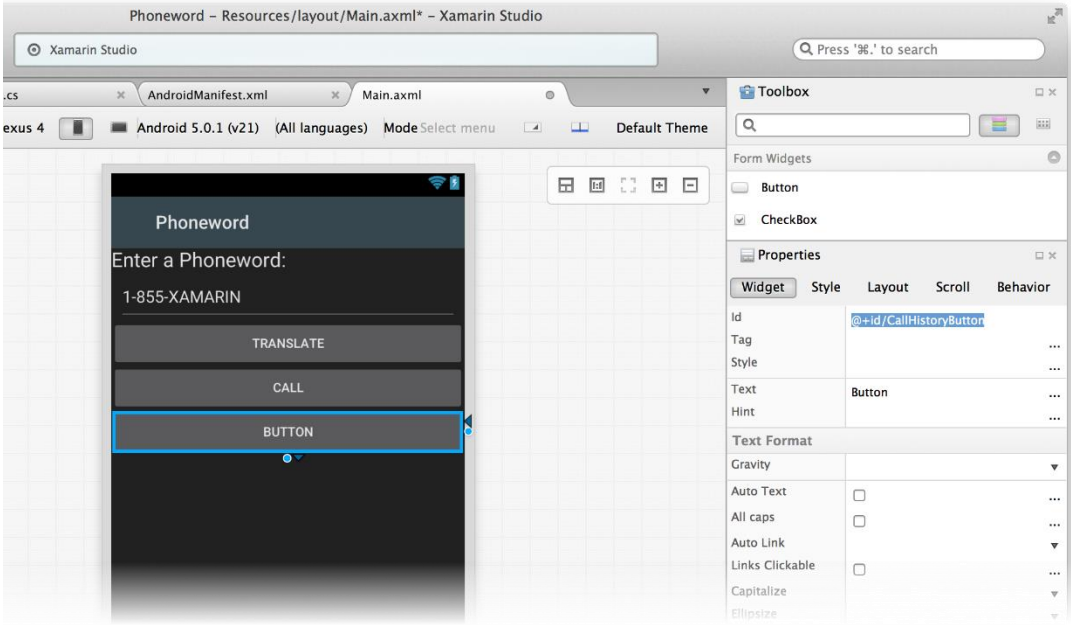
2. Let's start by editing the user interface. Open the **Main.axml** file from the *Solution Pad*:



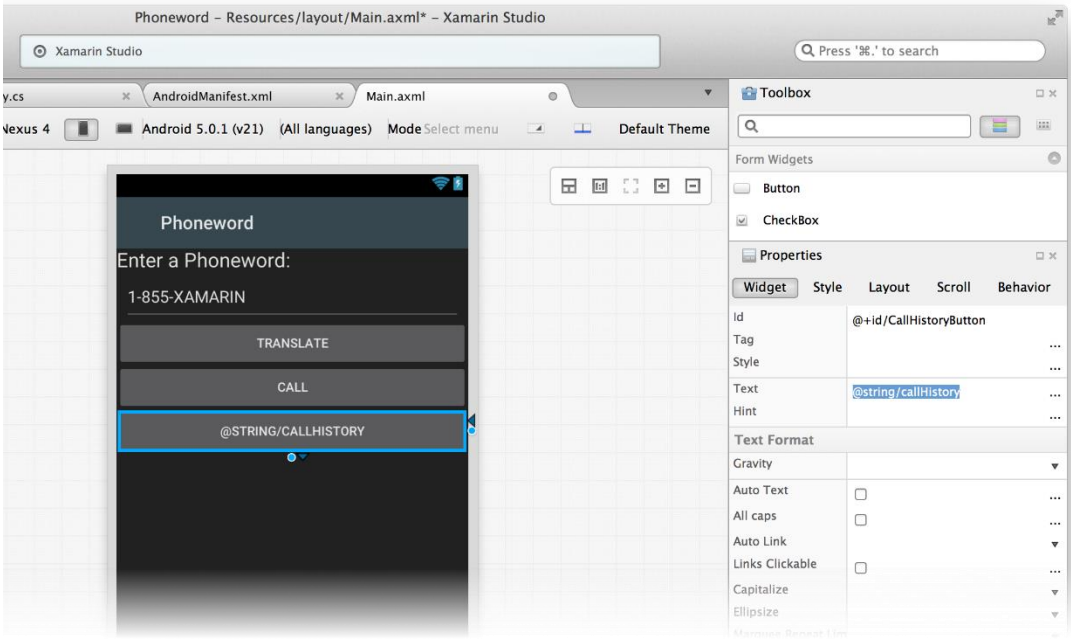
3. From the *Toolbox*, drag a *Button* onto the design surface and place it below the *Call* button:



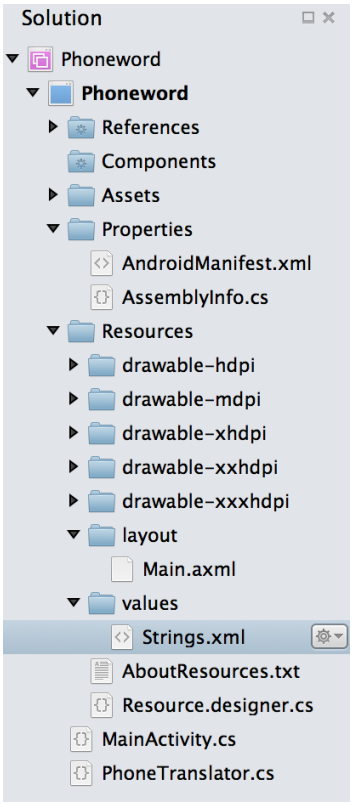
4. In the *Properties Pad*, change the button *Id* to `@+id/CallHistoryButton`:



5. Let's set the *Text* property of the button to `@string/callHistory`. The Android Designer will interpret this literally, but we're going to make a few changes so that the button's text shows up correctly:



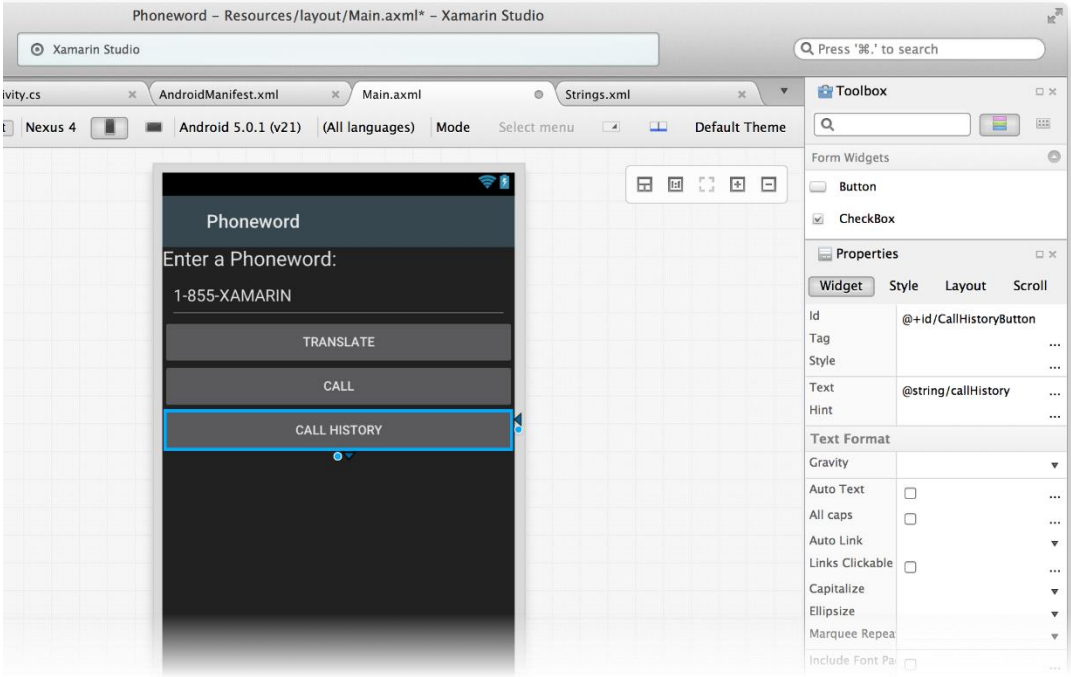
6. Expand the values node under the *Resources* folder in the *Solution Pad* and double-click the string resources file, **Strings.xml**:



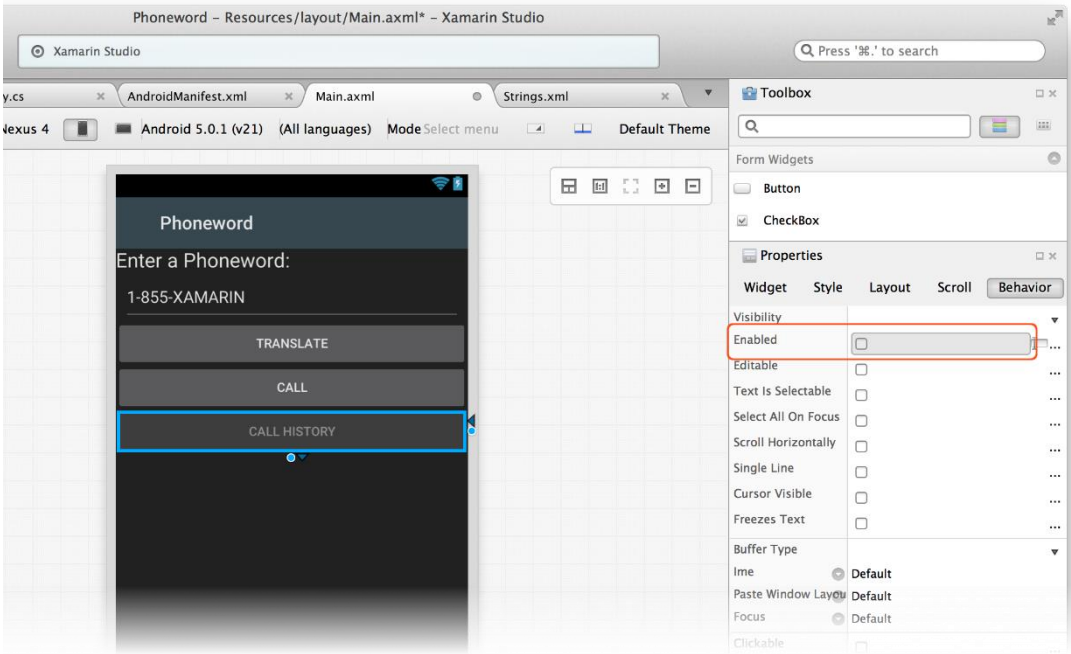
7. Add the `callHistory` string name and value to the **Strings.xml** file and save it:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="callHistory">Call History</string>
</resources>
```

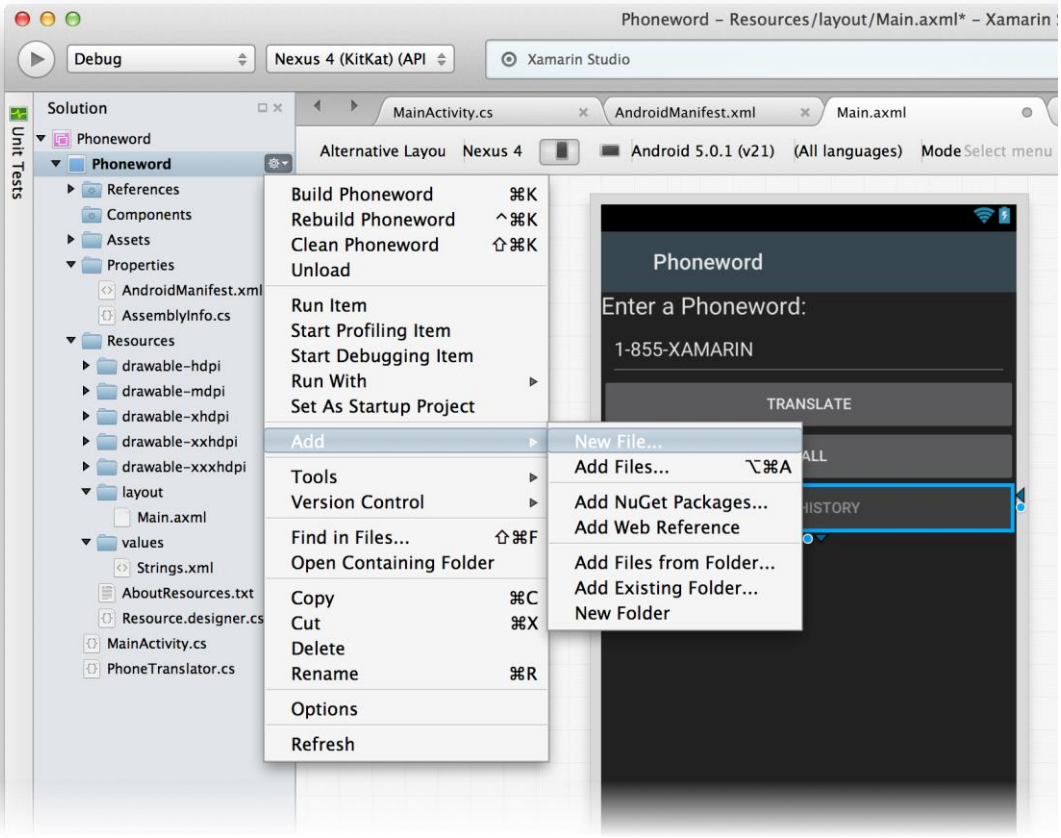
The *Call History* button text should update to reflect the new string value:



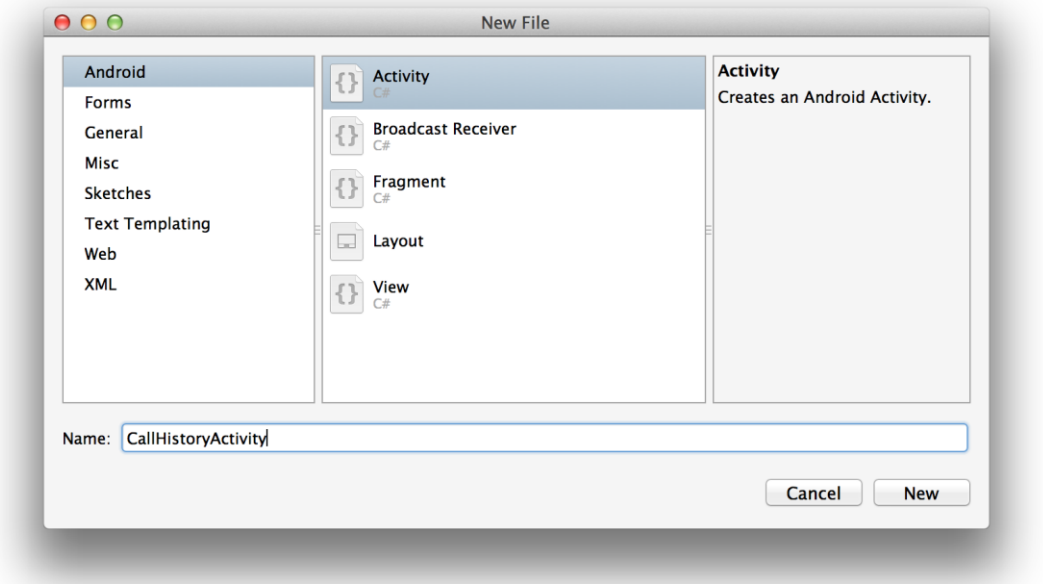
8. With the *Call History* button selected on the design surface, open the *Behavior* tab in the *Properties Pad* and double-click the *Enabled* checkbox to disable the button. This will cause the button to become darker on the design surface:



9. Let's create a second Activity to power the second screen. In the *Solution Pad*, click the gray gear icon next to the **Phoneword** project and choose *Add > New File...*:



10. From the *New File* dialog, choose *Android* > *Activity* and name the Activity `CallHistoryActivity`:



11. Replace the template code in `CallHistoryActivity` with the following:

```
using System;
```

```
using System.Collections.Generic;

using Android.App;

using Android.OS;

using Android.Widget;

namespace Phoneword

{

    [Activity(Label = "@string/callHistory")]

    public class CallHistoryActivity : ListActivity

    {

        protected override void OnCreate(Bundle bundle)

        {

            base.OnCreate(bundle);

            // Create your application here

            var phoneNumbers =
Intent.Extras.GetStringArrayList("phone_numbers") ?? new string[0];

            this.ListAdapter = new ArrayAdapter<string>(this,
Android.Resource.Layout.SimpleListItem1, phoneNumbers);

        }

    }
}
```

In this class, we're creating a `ListActivity` and populating it programmatically, so we don't need to create a new layout file for this Activity. We'll discuss this in more detail in the Hello, Android Multiscreen Deep Dive .

12. In our app, we're going to collect phone numbers that the user has dialed on the first screen and then pass them to the second screen. We're going to store the phone numbers as a list of strings. To support lists, add the following `using` directive to the top of the `MainActivity` class:

```
using System.Collections.Generic;
```

Next, let's create an empty list that we can fill with phone numbers. Our `MainActivity` class will look like this:

```
[Activity(Label = "Phoneword", MainLauncher = true, Icon =
"@drawable/icon")]
```

```
public class MainActivity : Activity
{
    static readonly List<string> phoneNumbers = new List<string>();
    ...// OnCreate, etc.
}
```

13. Let's wire up the *Call History* button. In the `MainActivity` class, add the following code to register and wire up the 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);
};
```

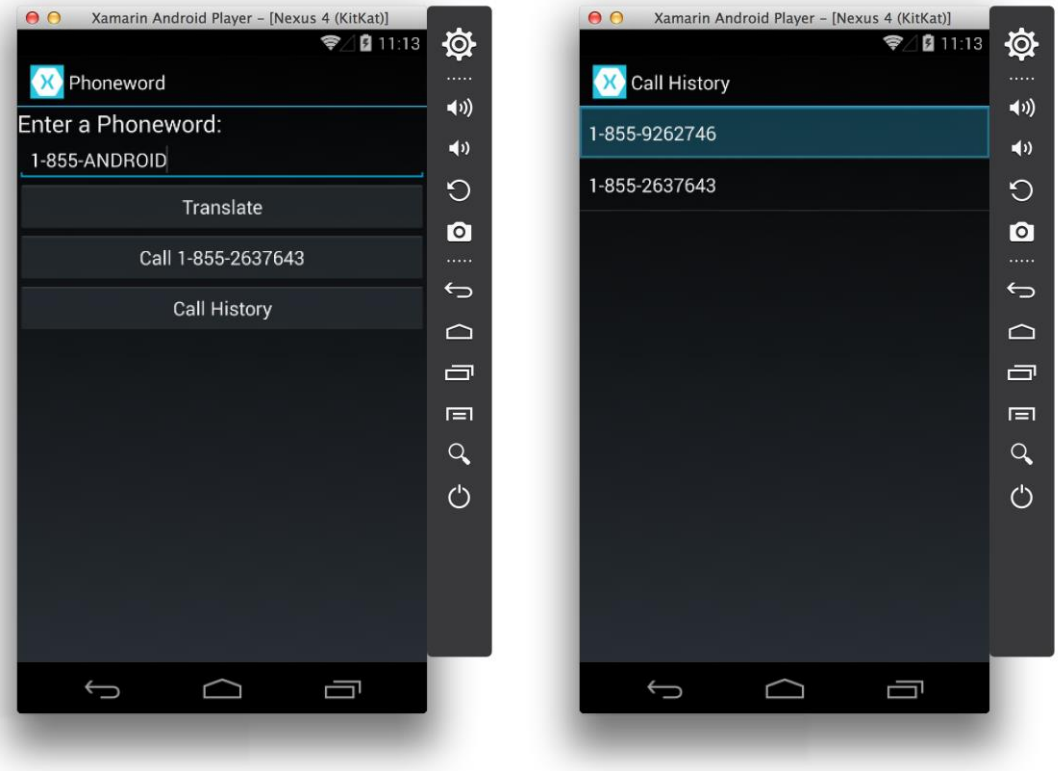
14. We want to extend the *Call* button's functionality to add a phone number to the list of numbers and enable the *Call History* button whenever the user dials a new number. Let's change the code of the ***Neutral Button*** in our Alert Dialog to reflect that:

```
callDialog.SetNeutralButton("Call", delegate
{
    // add dialed number to list of called numbers.
    phoneNumbers.Add(translatedNumber);
    // enable the Call History button
    callHistoryButton.Enabled = true;
    // Create intent to dial phone
    var callIntent = new Intent(Intent.ActionCall);
    callIntent.SetData(Android.Net.Uri.Parse("tel:" +
translatedNumber));
    StartActivity(callIntent);
```

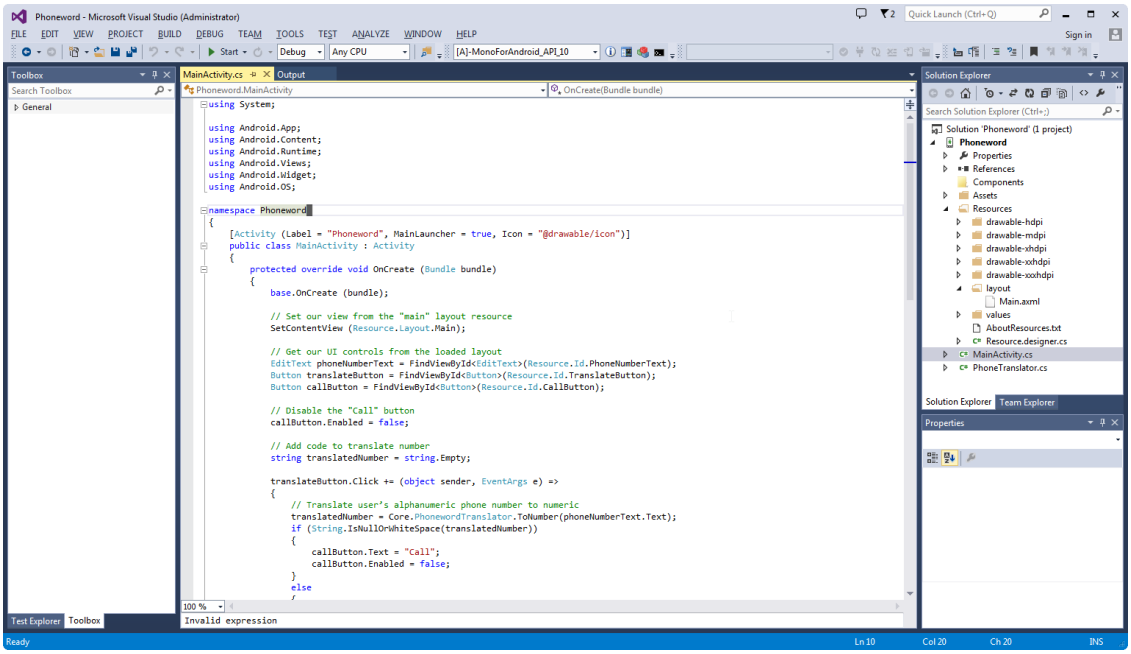
```
});
```

Save and build the application to make sure there are no errors.

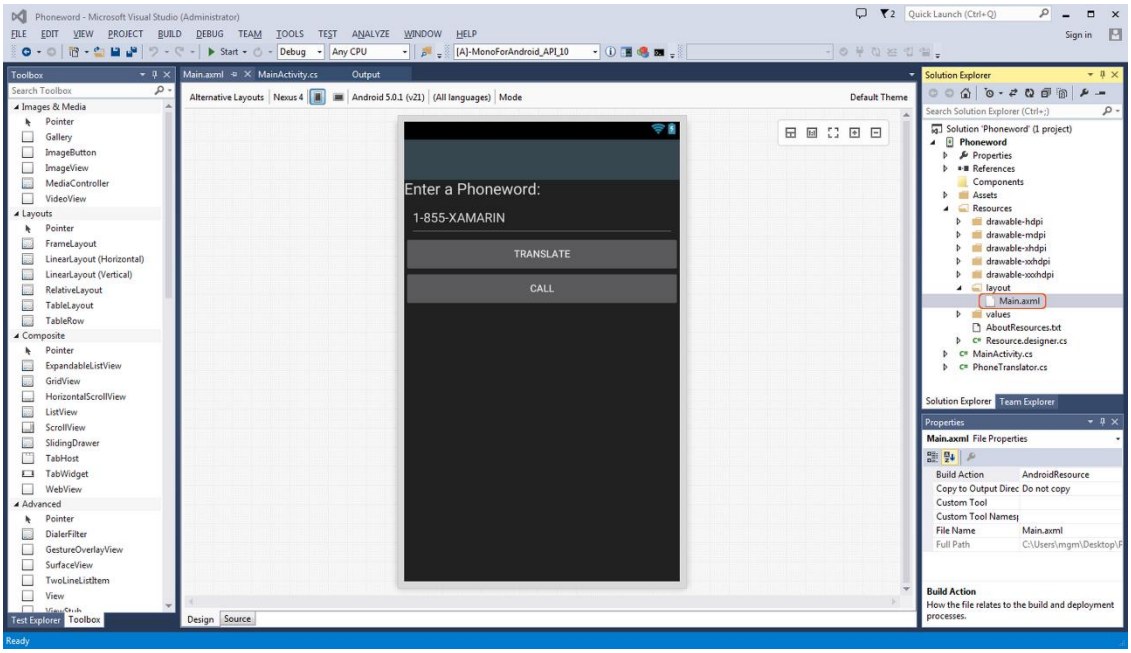
15. Deploy the application to an emulator or device. The following screenshots illustrate the *Phoneword* application running in the Xamarin Android Player:



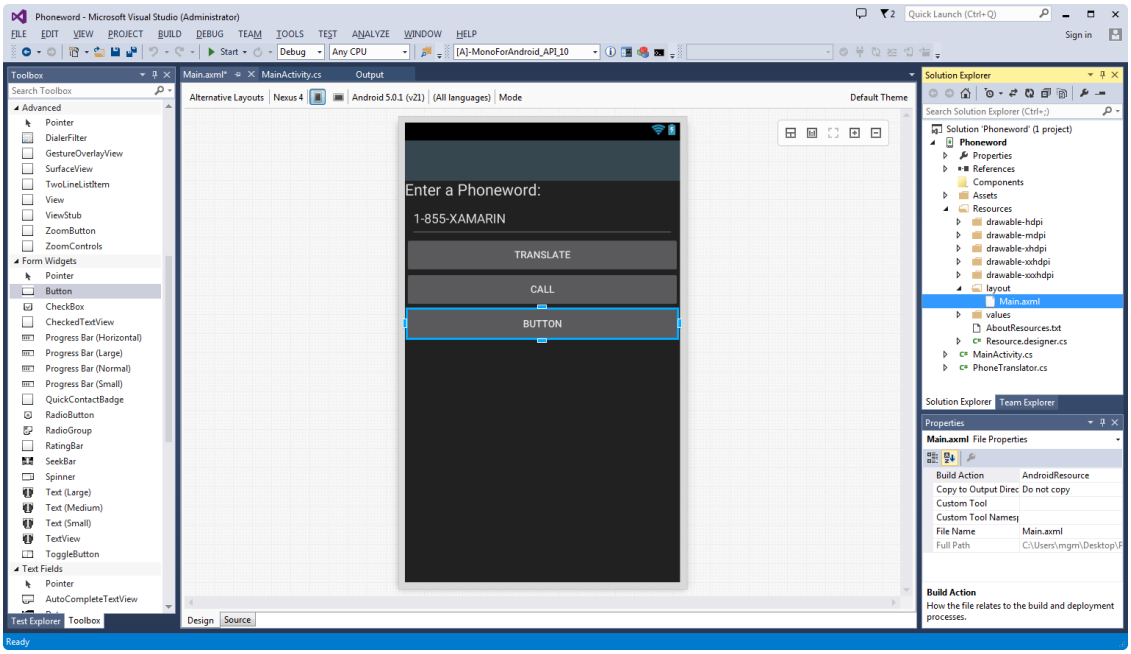
1. Let’s open the *Phoneword* application in Visual Studio:



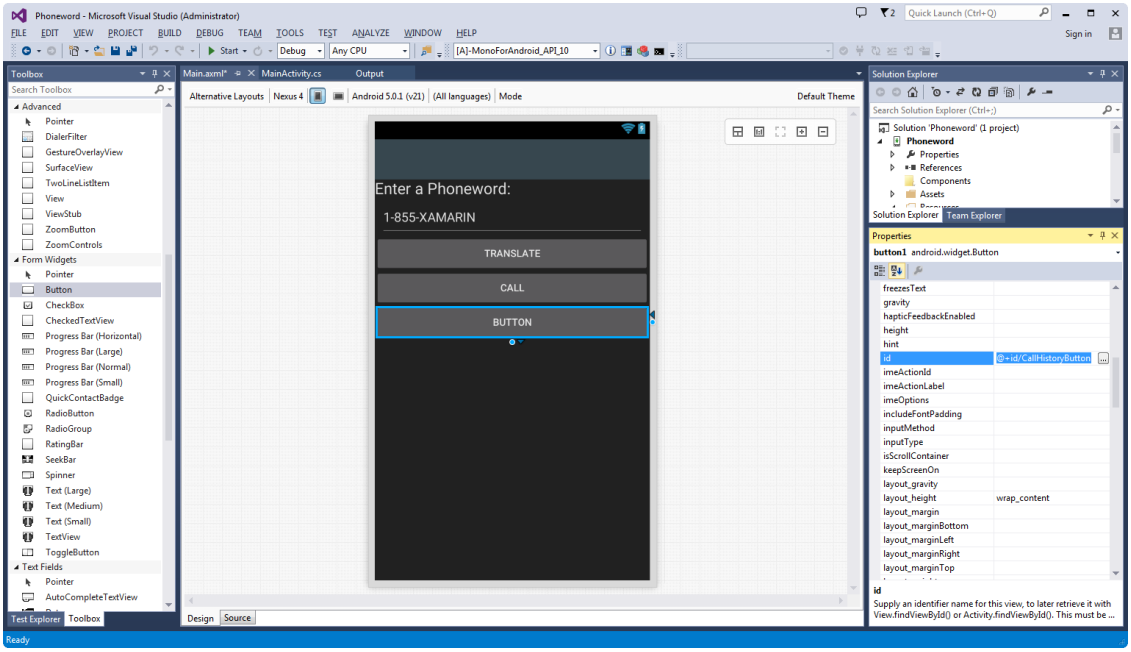
2. Let's start by editing the user interface. Open the **Main.axml** file from the *Solution Explorer*:



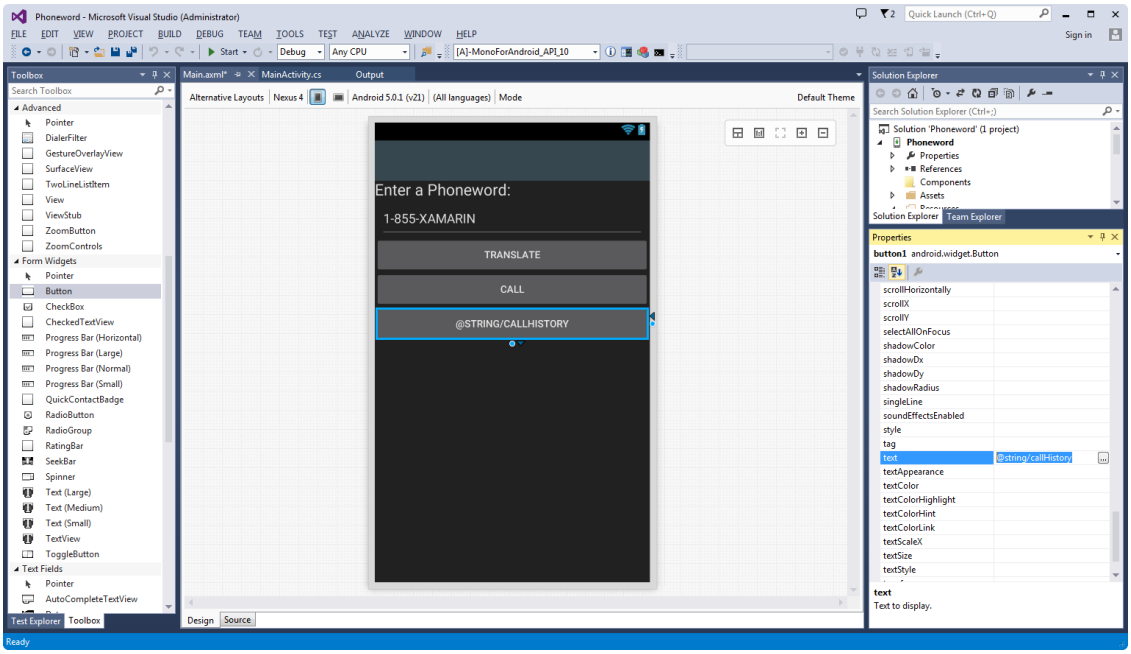
3. From the *Toolbox*, drag a *Button* onto the design surface and place it below the *Call* button:



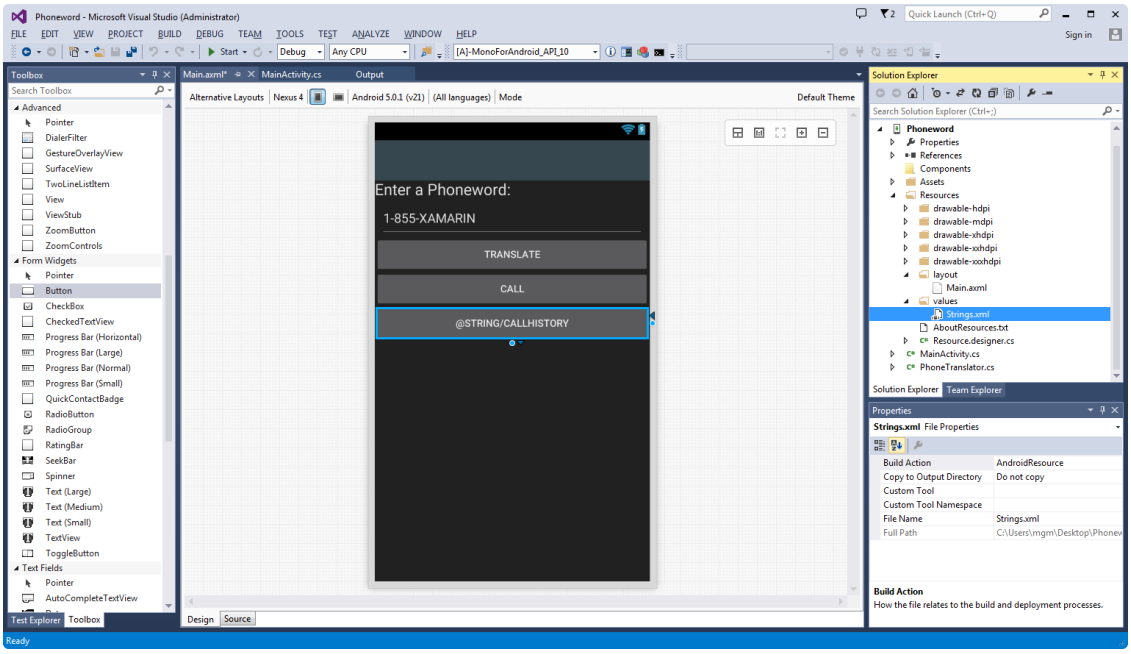
4. In the *Properties* pane, change the button *Id* to `@+id/CallHistoryButton`:



5. Let's set the *Text* property of the button to `@string/callHistory`. The Android Designer will interpret this literally, but we're going to make a few changes so that the button's text shows up correctly:



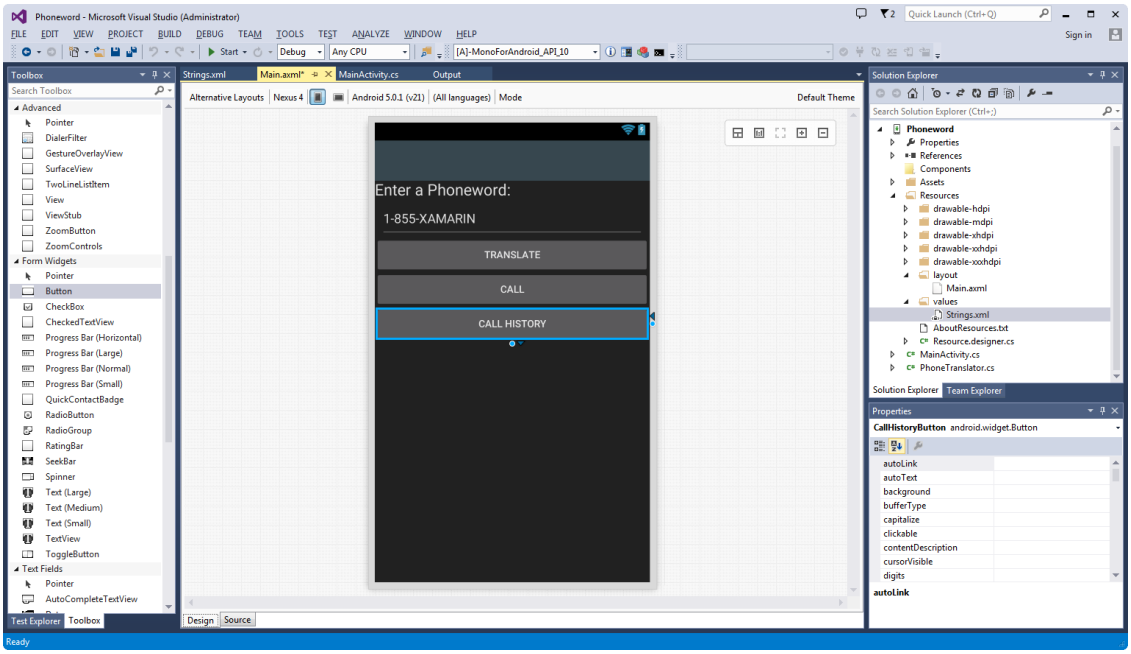
6. Expand the *values* node under the *Resources* folder in the *Solution Explorer* and double-click the string resources file, **Strings.xml**:



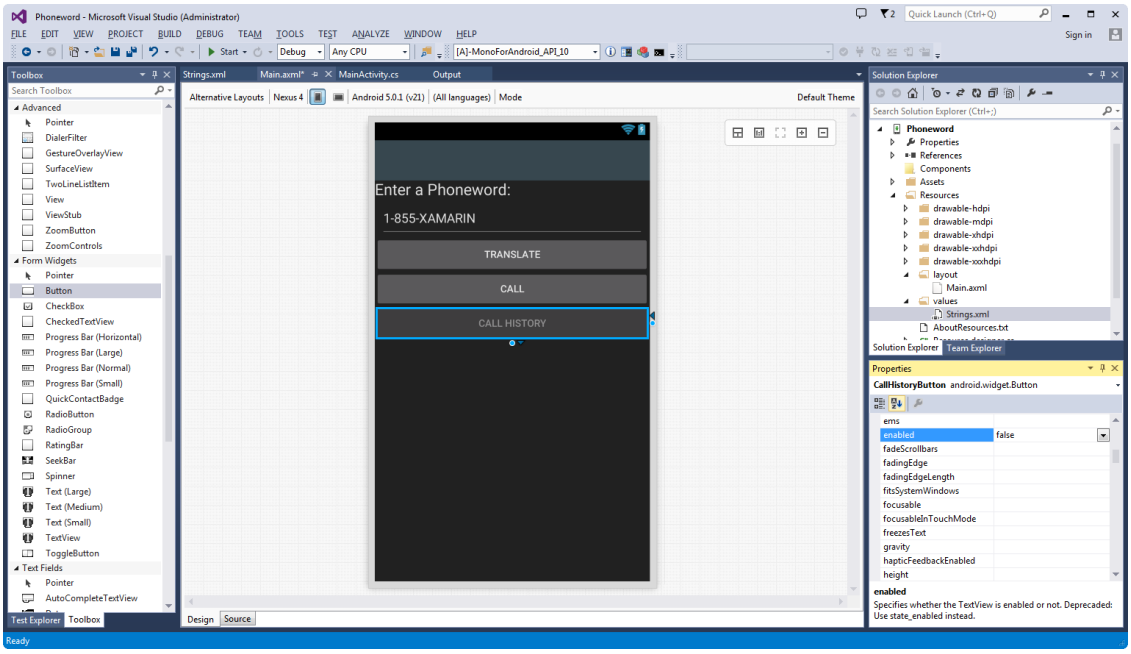
7. Add the `callHistory` string name and value to the **Strings.xml** file and save it:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="callHistory">Call History</string>
</resources>
```

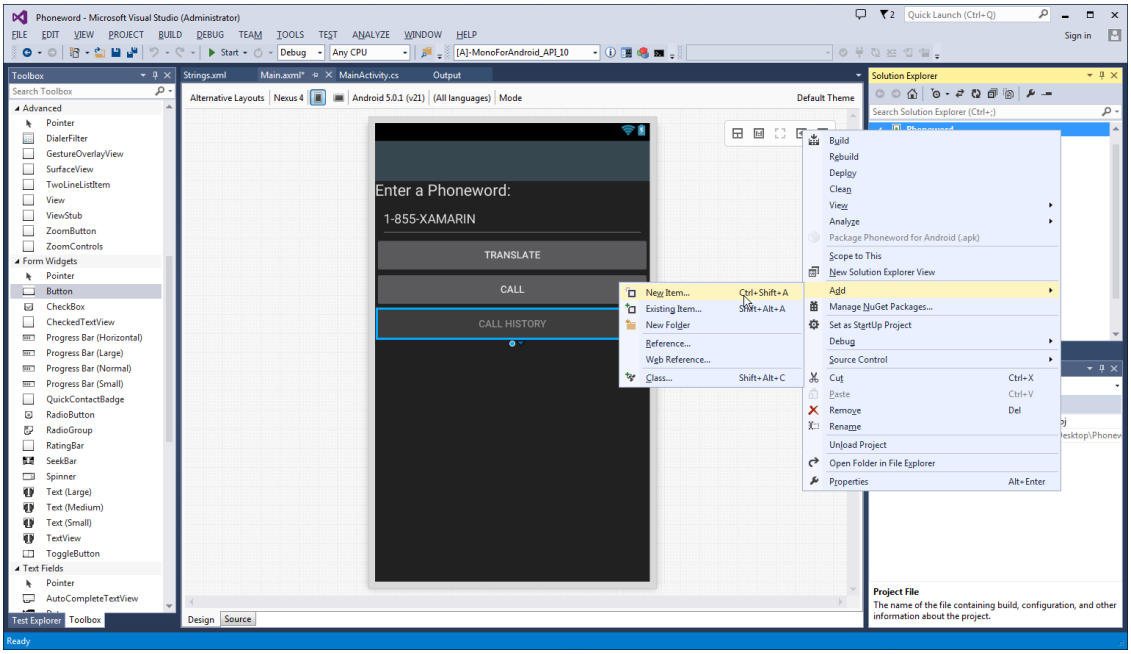
The *Call History* button text should update to reflect the new string value:



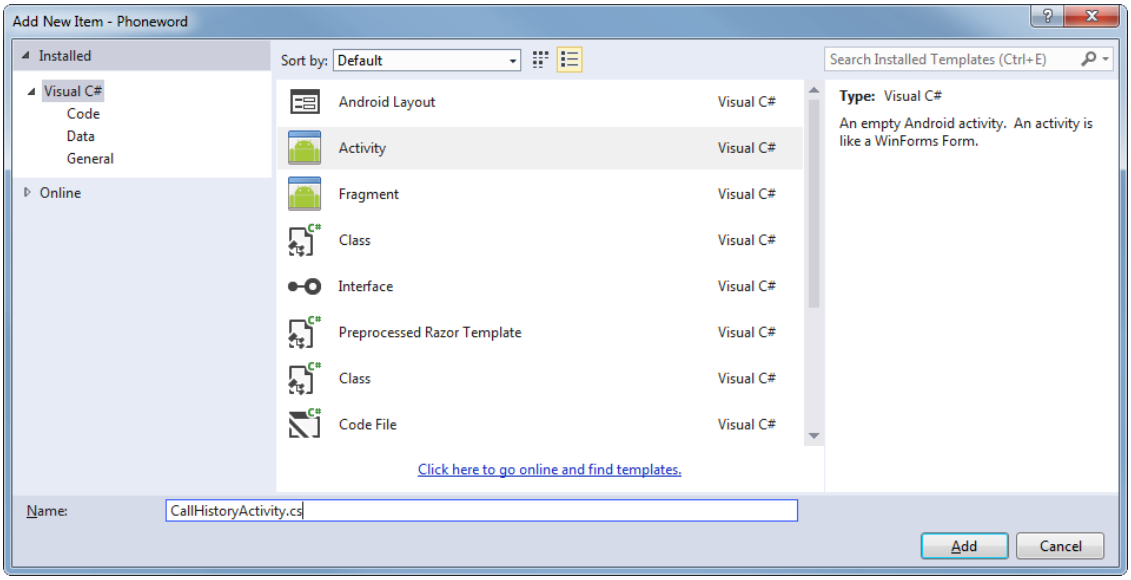
8. With the *Call History* button selected on the design surface, find the `enabled` setting in the *Properties* pane and set its value to `false` to disable the button. This will cause the button to become darker on the design surface:



9. Let's create a second Activity to power the second screen. In the *Solution Explorer*, right-click the **Phoneword** project and choose *Add* > *New Item...*:



10. In the *Add New Item* dialog, choose *Visual C# > Activity* and name the Activity file **CallHistoryActivity.cs**:



11. Replace the template code in **CallHistoryActivity.cs** with the following:

```
using System;

using System.Collections.Generic;

using Android.App;

using Android.OS;

using Android.Widget;

namespace Phoneword
```

```
{
    [Activity(Label = "@string/callHistory")]
    public class CallHistoryActivity : ListActivity
    {
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            // Create your application here
            var phoneNumbers =
Intent.Extras.GetStringArrayList("phone_numbers") ?? new string[0];
            this.ListAdapter = new ArrayAdapter<string>(this,
Android.Resource.Layout.SimpleListItem1, phoneNumbers);
        }
    }
}
```

In this class, we're creating a `ListActivity` and populating it programmatically, so we don't need to create a new layout file for this Activity. We'll discuss this in more detail in the Hello, Android Multiscreen Deep Dive.

12. In our app, we're going to collect phone numbers that the user has dialed on the first screen and then pass them to the second screen. We're going to store the phone numbers as a list of strings. To support lists, add the following `using` directive to the top of the `MainActivity` class:

```
using System.Collections.Generic;
```

Next, let's create an empty list that we can fill with phone numbers. Our `MainActivity` class will look like this:

```
[Activity(Label = "Phoneword", MainLauncher = true, Icon =
"@drawable/icon")]
public class MainActivity : Activity
{
    static readonly List<string> phoneNumbers = new List<string>();
    ...// OnCreate, etc.
```

```
    }
```

13. Let's wire up the *Call History* button. In the `MainActivity` class, add the following code to register and wire up the button:

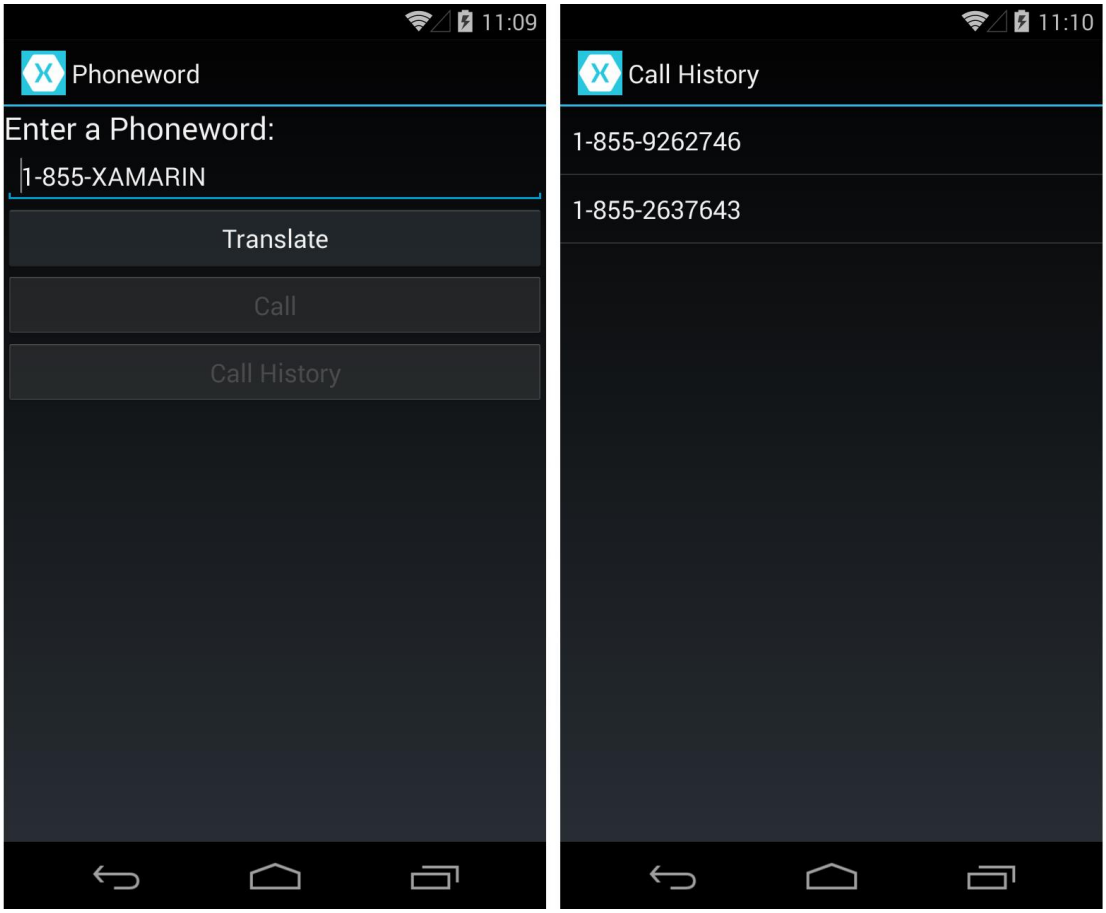
```
Button callHistoryButton = FindViewById<Button>
(Resource.Id.CallHistoryButton);

callHistoryButton.Click += (sender, e) =>

{

    var intent = new Intent(this, typeof(CallHistoryActivity));

    intent.PutStringArrayListExtra("phone_numbers", phoneNumbers);

    StartActivity(intent);

};
```

14. We want to extend the *Call* button's functionality to add a phone number to the list of numbers and enable the *Call History* button whenever the user dials a new number. Let's change the code of the ***Neutral Button*** in our Alert Dialog to reflect that:

```
callDialog.SetNeutralButton("Call", delegate

{

    // add dialed number to list of called numbers.

    phoneNumbers.Add(translatedNumber);

    // enable the Call History button

    callHistoryButton.Enabled = true;

    // Create intent to dial phone

    var callIntent = new Intent(Intent.ActionCall);

    callIntent.SetData(Android.Net.Uri.Parse("tel:" +
translatedNumber));

    StartActivity(callIntent);

});
```

Save and build the application to make sure there are no errors.

15. Deploy the application to an emulator or device. The following screenshots illustrate the *Phoneword* application running in the Xamarin Android Player:



Congratulations on completing your first multi-screen Xamarin.Android application! Now it's time to dissect the tools and skills we just learned in the Hello, Android Multiscreen Deep Dive.