Lab 01: Introduction to Cross Platform Development

Prerequisites

You will need a development environment, either a Mac or Windows PC with the Android SDK, iOS SDK and Xamarin tools installed. We will be using the Android and iOS emulators to test the code we are building, so make sure to have a virtual device already configured and ready to run. See the Cross-Platform Installation Guild if you need help getting your environment setup.

Downloads

Included with this lab document is a folder with resources that you will need in order to complete the lab. The folder name is **Fundamentals Introduction to Cross Platform Resources**. Make sure you have this folder before you begin.

Lab Goals

The goal of this lab is to introduce the Xamarin platform and how to architect a cross-platform application to maximize code re-use and deliver a high-quality native experience on iOS and Android.

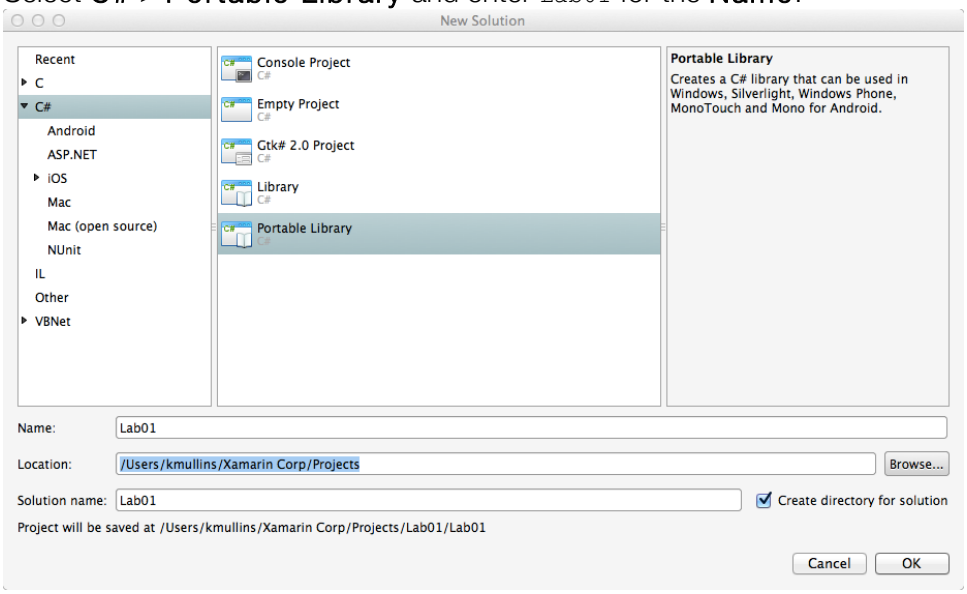
In this lab you will be creating a new Portable Class Library (PCL) solution, making changes to small blocks of code, and adding platform specific projects to the solution to consume the PCL as you follow along with the instructor.

Steps

Creating the PCL Solution

1. Launch **Xamarin Studio** using Spotlight or the application icon.
2. Click on **File** > **New...** > **Solution...**

3. Select **C#** > **Portable Library** and enter `Lab01` for the **Name**:

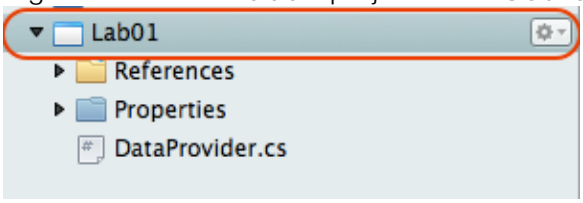


4. Click the **OK** button finish creating the solution.

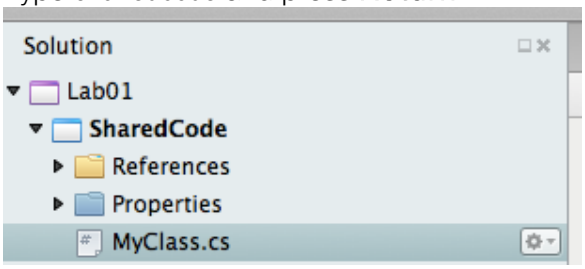
Rename the PCL Project

To make the structure of our solution easier to understand and maintain we are going to change the default name of the PCL project.

1. Right click on the **Lab01** project in the **Source Tree** and select **Rename**:



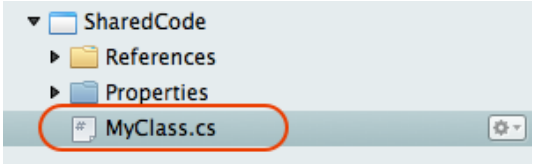
2. Type `SharedCode` and press **Return**:



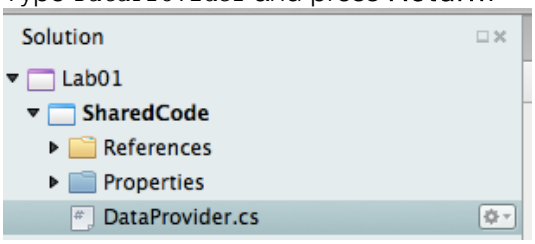
Rename the Default Class

Again, to make the structure of our solution easier to understand and maintain, we are going to rename the default class that was automatically added to our PCL project.

1. Right click on the **MyClass.cs** class in the **Source Tree** and select **Rename**:



2. Type `DataProvider` and press **Return**:



Create the Data Provider

Next we are going to create a data provider that will provide shared data across our platform specific mobile application interface projects.

1. Double click on the **DataProvider.cs** class in the **Source Tree** to open it for editing:

2. Edit the file and make it look like the following code:

```
using System;

namespace Lab01
{
   public class DataProvider
   {
      public DataProvider ()
      {
      }

      public static string[] GetRecords() {

         string[] items = new string[] {"Miguel de Icaza", "Nat Friedman", "Bart Decr
em", "Scott Hanselman"};

         return items;
      }
```

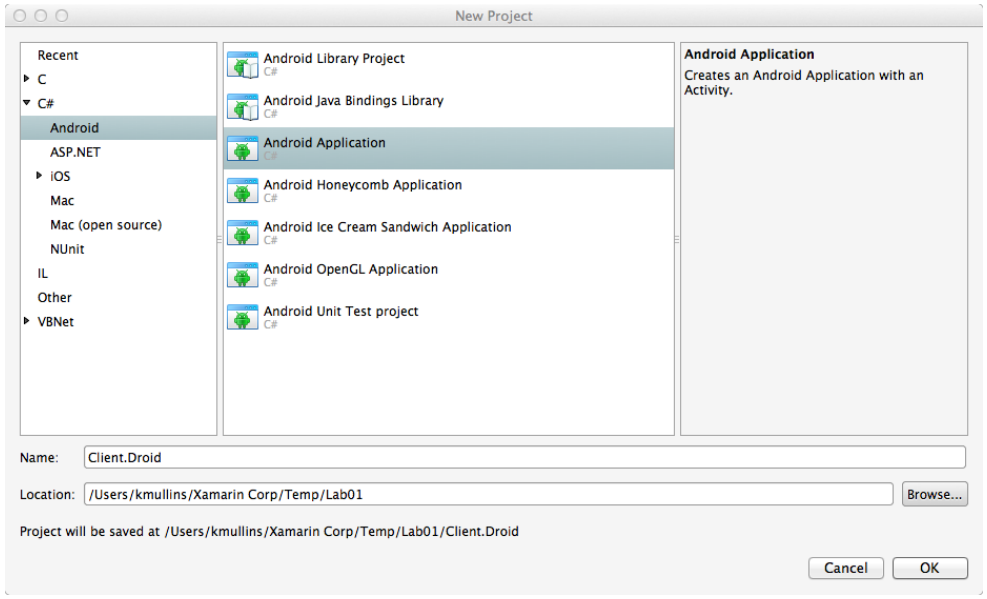
```
    }
}
```

3. **Save** the project.
4. **Build** the project and ensure that there are no errors.

Adding an Android Project

Now we are going to add simple Android specific mobile application to the solution to consume the data provider that we created in the PCL.

1. Right click on the **Lab01** solution in the **Source Tree** and select **Add New Project...**
2. Select **C#** > **Android** > **Android Application** and enter `Client.Droid` for the **Name**:

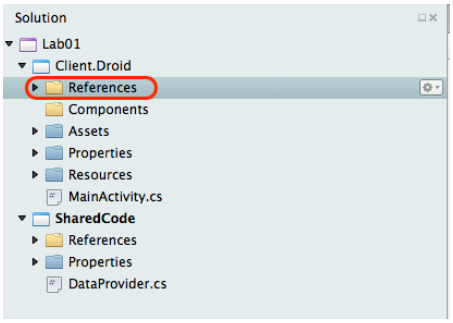


3. Click OK to finish creating the new project.

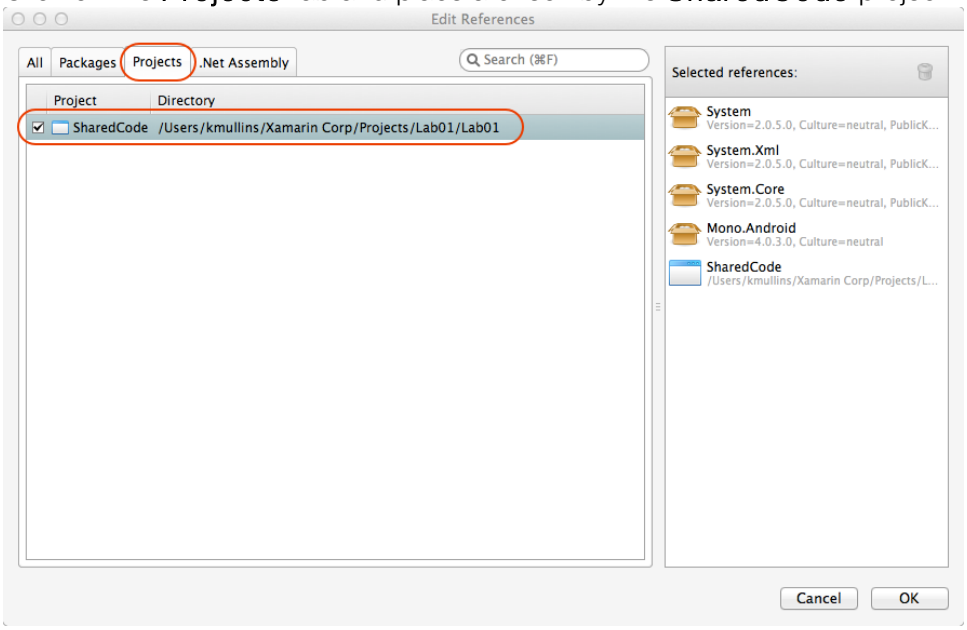
Adding a Reference to the PCL Project

Next we need to add a reference to the PCL project so that the Android project can access the shared data provider we created above.

1. Select the **References** folder under the **Client.Droid** project in the **Source Tree**:



2. Right click on **References** and select **Edit References...**
3. Click on the **Projects** tab and place a check by the **SharedCode** project:

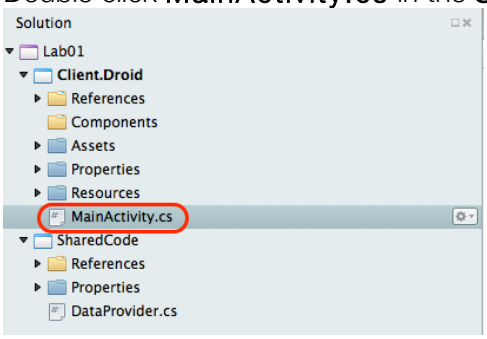


4. Click the **OK** button to add the reference and close the dialog box.

Consuming the PCL

We are going to modify the default button in our Android project to consume the shared data provided by the PCL and cycle through each item when the user clicks the button.

1. Double click **MainActivity.cs** in the **Source Tree** to open it for editing:



2. Add the following Using statement to the top of the file:

```
Using Lab01;
```

3. Edit the MainActivity class and make the code look like the following:

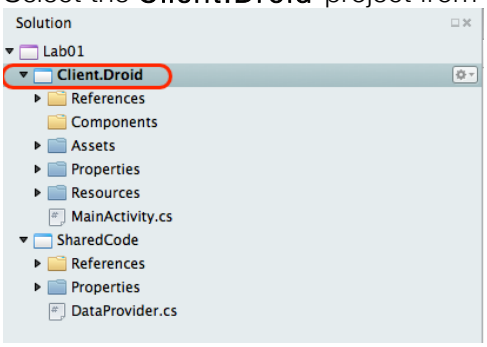
```
public class MainActivity : Activity
    {
      private int index = 0;
      private string[] item=DataProvider.GetRecords();

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

        SetContentView (Resource.Layout.Main);

        Button button = FindViewById<Button> (Resource.Id.myButton);

        button.Click += delegate {
          button.Text = string.Format ("Speaker {0} - {1}", index+1, item[index]);

          if (++index>=item.Length ) index=0;
        };
      }
    }
```

4. **Save** the project.
5. **Build** the project and ensure that there are no errors.

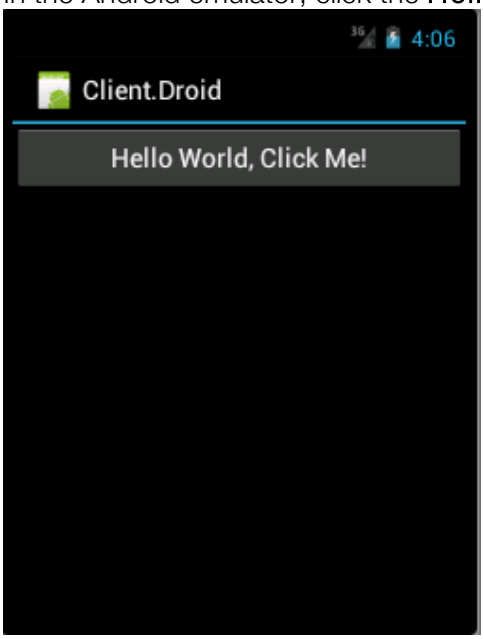
Testing the Android Application

Next we will test our Android application and assure that it is consuming the shared data provided by the data provider in our PCL.

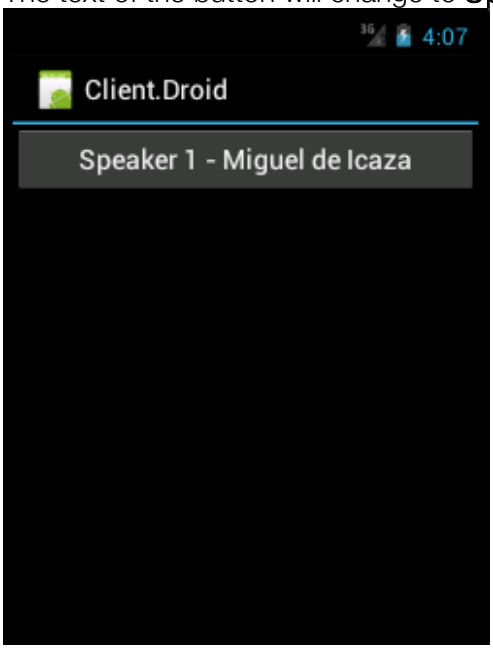
1. Select the **Client.Droid** project from the **Source Tree**:



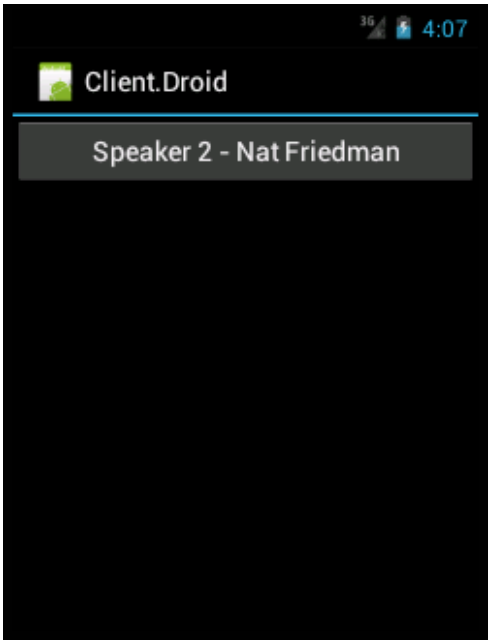
2. Right click on **Client.Droid** and select **Set As Startup Project**.
3. Click on the **Run** menu and select **Start Debugging**.
4. In the Android emulator, click the **Hello World, Click Me!** button:



5. The text of the button will change to **Speaker 1 – Miguel de Icaza**:



6. Click the button again and the text will change to **Speaker 2 – Nat Friedman**:

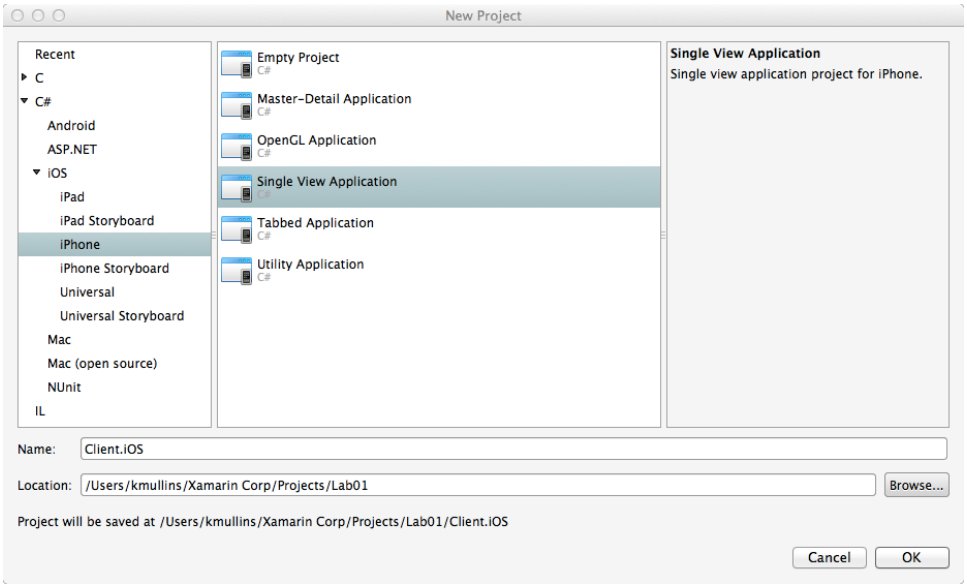


7. Continue to click the button until it cycles back around to the first instructor.
8. Stop debugging the application.

Adding an iOS Application

Like the Android example above, we are going to add simple iOS-specific application to the solution to consume the PCL as well.

1. Right click on the **Lab01** solution in the **Source Tree** and select **Add New Project...**
2. Select **C# > iOS > iPhone > Single View Application** and enter `Client.iOS` for the **Name**:

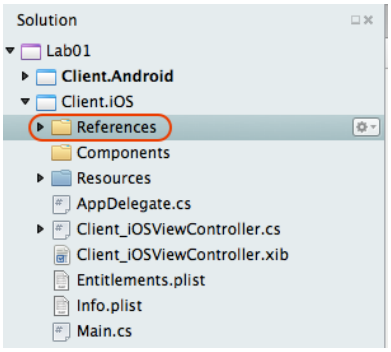


3. Click OK to finish creating the new project.

Adding a Reference to the PCL Project

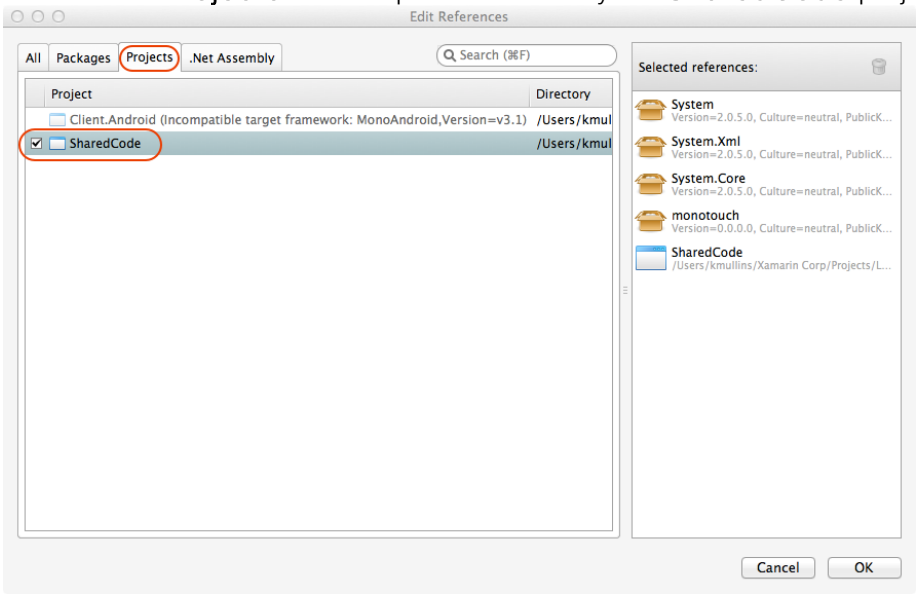
Next we need to add a reference to the PCL solution so that the iOS project can access the shared data provider we created above.

1. Select the **References** folder under the **Client.iOS** project in the **Source Tree**:



2. Right click on **References** and select **Edit References...**

3. Click on the **Projects** tab and place a check by the **SharedCode** project:

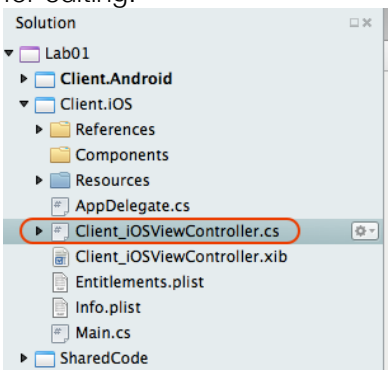


4. Click the **OK** button to add the reference and close the dialog box.

Consuming the PCL

We are going to add a button to our iOS project to consume the shared data provided by the PCL and cycle through each item when the user clicks the button.

1. Double click **Client_iOSViewController.cs** in the **Source Tree** to open it for editing:



2. Add the following Using statement to the top of the file:
   Using Lab01;

3. Add the following statements to the top of the **Client_iOSVierController** class:

```
private int index;
private string[] item=DataProvider.GetRecords();
```

4. Edit the **ViewDidLoad** class and make the code look like the following:

```
UIButton button = new UIButton (UIButtonType.RoundedRect);
button.Frame = new RectangleF (0, 50, View.Frame.Width, 32);
button.SetTitle ("Click Me!", UIControlState.Normal);
```

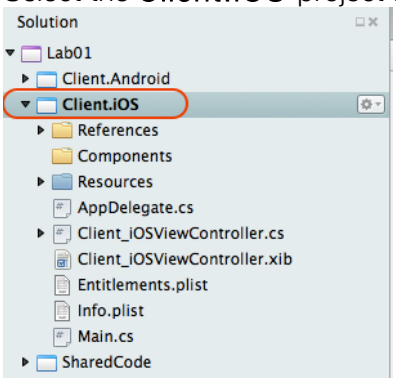
```
            View.AddSubview (button);

            button.TouchUpInside += (sender, e) => {
               button.SetTitle (string.Format ("Speaker {0} -
{1}", index+1, item[index]), UIControlState.Normal);

               if (++index>=item.Length ) index=0;
            };
```

5. **Save** the project.
6. **Build** the project and ensure that there are no errors.

Testing the iOS Mobile Application

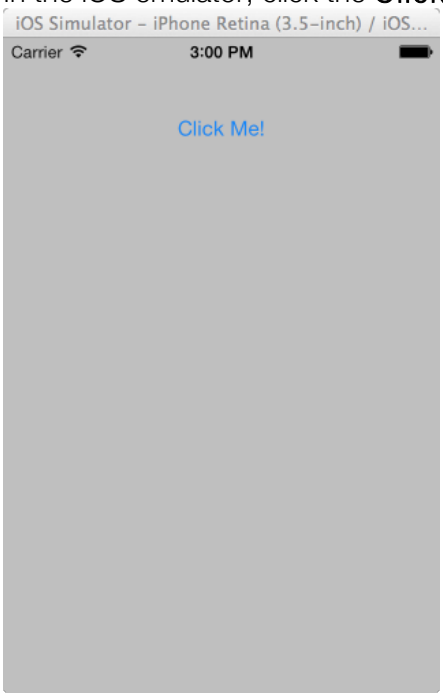
Next we will test our iOS mobile application and assure that it is consuming the shared data provided by the data provider in our PCL.

1. Select the **Client.iOS** project from the **Source Tree**:

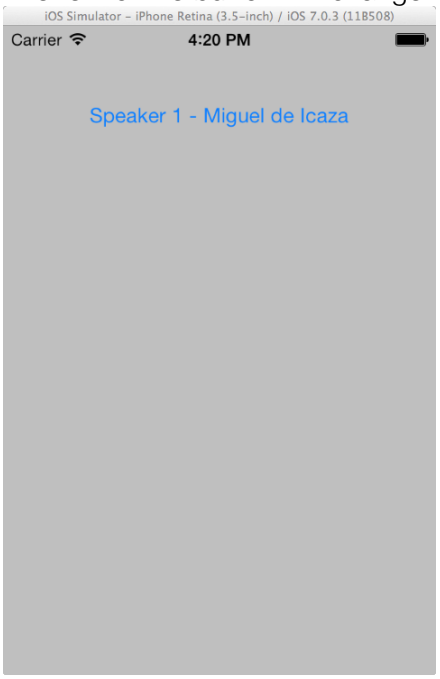


2. Right click on **Client.iOS** and select **Set As Startup Project**.
3. Click on the **Run** menu and select **Start Debugging**.

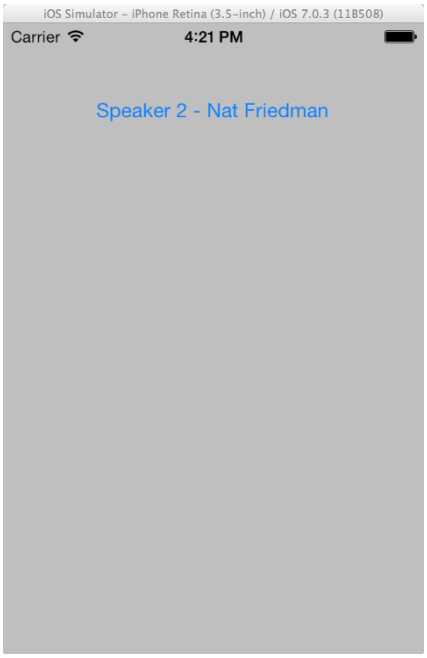
4. In the iOS emulator, click the **Click Me!** button:



5. The text of the button will change to **Speaker 1 – Miguel de Icaza**:



6. Click the button again and the text will change to **Speaker 2 – Nat Friedman**:



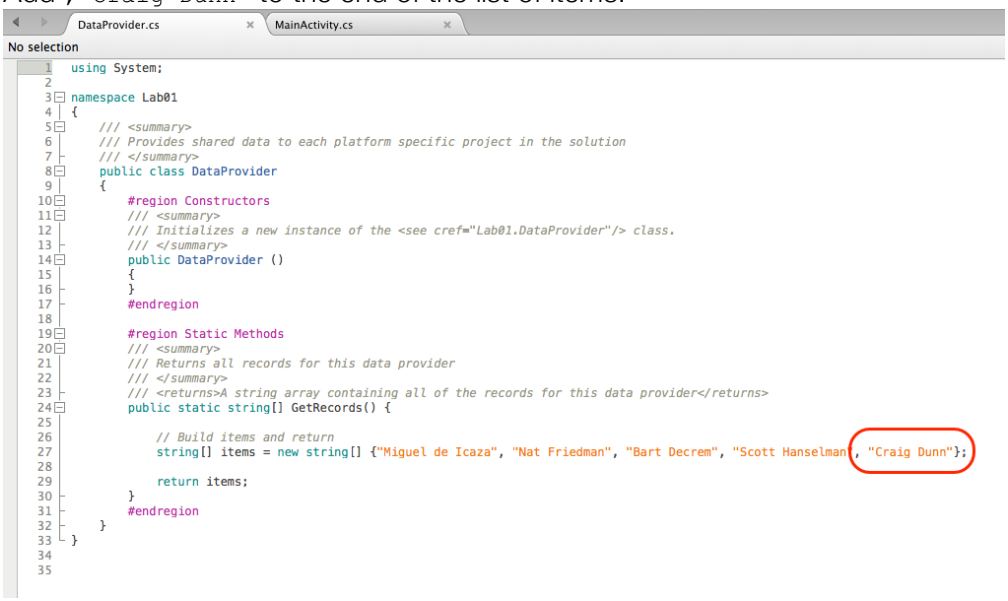
7. Continue to click the button until it cycles back around to the first instructor.
8. Stop debugging the application.

Modifying the PCL

Now we are going to modify the data provider in our PCL and add a new record.

1. Select **DataProviders.cs** from the **Source Tree** and double click it to open it for editing:

2. Add `,"Craig Dunn"` to the end of the list of items:

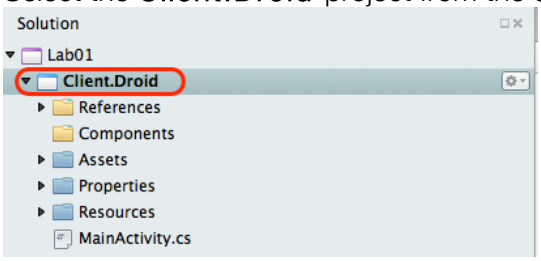


3. **Save** the project.
4. **Build** the project and ensure that there are no errors.
5. Let the instructor know if you have any trouble.

Testing the Change

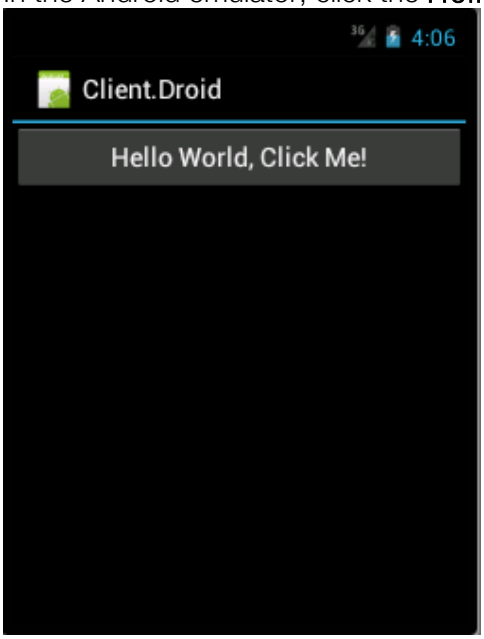
Now we are going to run both our Android and iOS project and make sure they show the new item we added to the data provider in our PCL.

1. Select the **Client.Droid** project from the **Source Tree**:

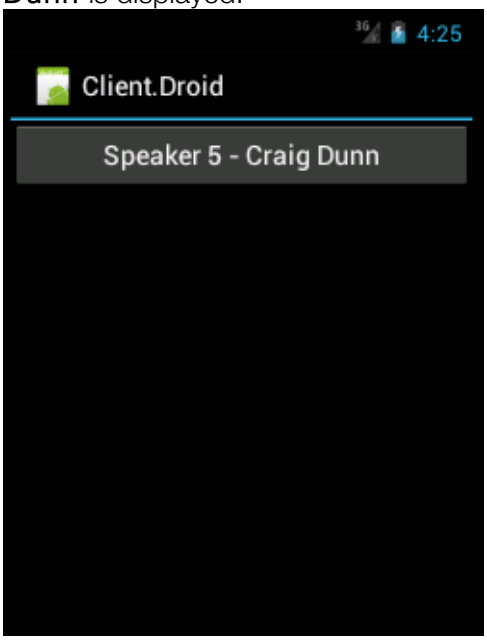


2. Right click on **Client.Droid** and select **Set As Startup Project**.
3. Click on the **Run** menu and select **Start Debugging**.

4. In the Android emulator, click the **Hello World, Click Me!** button:

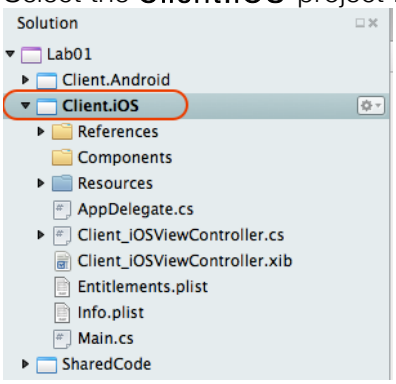


5. Keep clicking the button and cycle through the items until **Speaker 5 – Craig Dunn** is displayed:

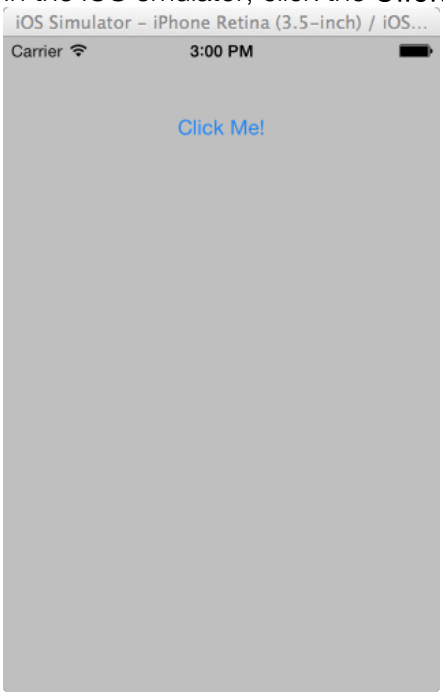


6. Stop debugging the application.

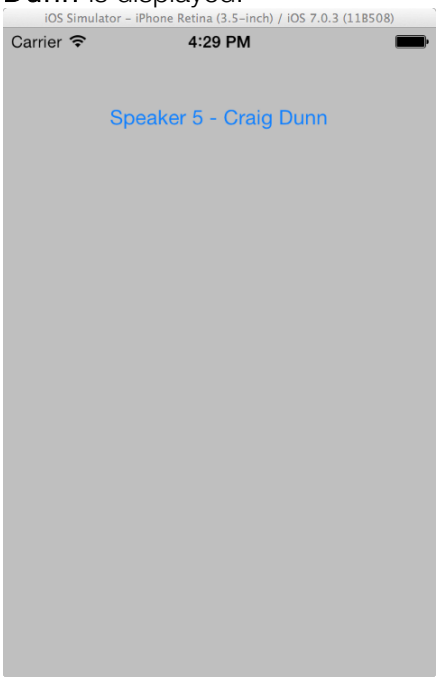
7. Select the **Client.iOS** project from the **Source Tree**:



8. Right click on **Client.iOS** and select **Set As Startup Project**.
9. Click on the **Run** menu and select **Start Debugging**.
10. In the iOS emulator, click the **Click Me!** button:



11. Keep clicking the button and cycle through the items until **Speaker 5 – Craig Dunn** is displayed:



12. Stop debugging the application.

Summary

In this lab, we created a new PCL solution, added functionality to the solution to share across platforms and added platform specific Android and iOS application projects to consume the shared functionality.