## FUNDAMENTALS LESSON (TEMPLATE)

## (Lesson Title)

INSTRUCTOR OVERVIEW

ESTIMATED TIME: 2 hours

LABS

* Lab 01 – create and run template app
* Lab 02 – edit template to build basic Phoneword app

## Prep

* Clear Xamarin Studio (projects, no open solution)
  + File > Recent Solutions > Clear
* Android Emulator is configured & running
  + x86 KitKat, reset content to remove apps, start it up, etc.
* Your computer
  + turn off notifications
  + stop syncing dropbox, etc.
* Restore default color theme, etc
* Ensure desktop or whatever areas will be visible to attendees is clean
* Stuff for this lesson?

## Slides 01-0x: Script

### Slide 01: Xamarin University

Welcome to Xamarin University.

My name is \_\_\_\_\_\_\_\_\_\_ and I'll be your instructor today. Today’s lesson is \_\_\_\_\_\_\_\_\_\_\_\_\_\_. The primary focus of our course is to learn how to develop with Xamarin/Visual Studio through instructor led labs. We will begin with a short discussion describing what we are going to do in the lab and then we'll work through the labs together.

As I lead you through the steps of \_\_\_\_\_\_\_\_\_\_\_\_\_\_, I'll be explaining Xamarin/Visual Studio and what you need to know about the app we are building.

I'll do my best to lead at a speed comfortable for everyone. It is very important for everyone to follow along with the steps, so please interrupt me if you have a question or if you need more time to complete a lab step.

The microphones will all be on so we can have a very interactive class.

Before we begin I want to mention the resources available to you after class is over. You'll have all of the code and resource files we used to build the labs today including completed versions of the labs.

In addition, there is also supporting documentation explaining background concepts for the labs like \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, and so on.

### Slide 02: Module Objectives

In our first lab, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

## Slide 0x, Lab 01

INSTRUCTOR TASKS FOR THIS LAB (DO NOT READ ALOUD)

Create PhonewordAndroid project using Android Application template.

Place it on the Desktop

Build it and show the Select Emulator screen.

Start the emulator (if it’s not already) and run the application.

Explain Xamarin Studio and Project setup

### Project Structure Explanation

INSTRUCTOR:

Review each of these items, highlighting them in the IDE while you explain – use the following table as a guide to ensure you hit all the talking points.

Now let’s look at some of the nodes in the solution pane for our Xamarin.Android project:

|  |  |
| --- | --- |
| **References Folder** | Contains **.NET assemblies** needed to build and run this program |
| **Components** | Contains third-party and Xamarin components installed from the **Xamarin Components store**. |
| **Assets Folder** | Any raw data files necessary to run the application should be placed here (such as default database file or XML data files). They will be copied into the deployment package so they are available at runtime through the Android AssetManager.  Note: Make sure to open the AboutAssets.txt file as it has instructions on how to use the assets at runtime. |

INSTRUCTOR:

Open each of these files in the IDE while you explain.

The project template has already included a C# file and an XML layout file:

|  |  |
| --- | --- |
| **MainActivity.cs** | This contains the main activity for the application. Activities represent screens in Android apps, each unique screen shown to a user will be created from an Activity. |
| **Main.axml** | this is the designer file that creates the initial UI that is used by MainActivity.cs. It is an XML file which can be hand-edited, or edited with the built in Xamarin Studio designer. This is what we will be editing as we make UI changes |

INSTRUCTOR:

Run the app in the Android emulator

Explain that this is the template app.

Explain how to start the emulator

### Create the new UI

Now we’re going to edit the layout…

|  |  |
| --- | --- |
| INSTRUCTOR:  Open the /Resources/Layout/Main.axml file – should open in the Xamarin UI Designer | |
| Drag **TextView (LARGE)** into layout | Double-click to edit display label Enter a Phoneword: |
| Drag EditText into layout | Double-click to edit display label 1-855-xamarin |
|  | Use Properties panel to set ID PhoneNumberText |
| Drag Button into layout | Double-click to edit display label Translate |
| Drag Button into layout | Double-click to edit display label Call |
| Make sure to point out the +@id/NAME syntax used for identifying elements in code behind as you assign them to the buttons. | |
| Switch designer to XML view to demonstrate what the underlying file format looks like. | |

INSTRUCTOR:

Run the app in the Android emulator – highlight differences from editing the XML – new controls, but the buttons don’t DO anything.

### Add MainActivity Logic

|  |  |
| --- | --- |
| INSTRUCTOR:  Open the MainActivity.cs file | |
| Disable CallButton initially | CallButton.Enabled = false |
| Add existing PhoneWordTranslate.cs file to project | Should be in the assets folder for the lab. |
| Add Click handler to TranslateButton | TranslateButton.Click += … |
| Translate PhoneNumberText and add to CallButton (if not empty), |  |
| Enable CallButton |  |
| Add Click handler to CallButton, show AlertDialog, add Neutral and Negative buttons – use Intent.ActionCall to dial phone |  |

**Brief explanation of Intents**

Intents are used to ask the system to execute a specific behavior or operation. Often they will be used to present a new Activity with corresponding UI, but they can be used to signal non-UI behavior as well. We’ll use Intents to show a second screen in Lab2, and here we can use an Intent to ask the system to dial a number for us. We prefix the number with “tel:” to indicate the type of number we are dialing.

|  |
| --- |
| INSTRUCTOR:  RUN APP – NOTE SECURITY FAILURE |
| Explain security restrictions + sandboxing for Android, show permissions in manifest and request CALL permission. |

|  |
| --- |
| INSTRUCTOR:  RUN APP TO DIAL |

### Finishing Touches

|  |  |
| --- | --- |
| INSTRUCTOR:  Open the MainActivity.cs file | |
| Show the current state of icon and app name | [Activity] attribute |
| Change App name in options and on MainActivity | Right-=click |
| Add App icons | explain different resolution styles for drawable folder structure |

### Lab 01 Summary

Now you’ve seen how to build a basic single-screen Android application with the UI designer, provide behavior through an Activity, add in custom icons and title and debug and run the application with the Emulator and Xamarin Studio.

## Lab 02

* Open the starter project.
* Build it.

### Edit UI

CallHistoryButton > Text = @string/callHistory, [EXPLAIN STRING TABLES]

> Enabled = false

### Add new CallHistoryActivity

Now we’re going to build a new screen from scratch – using a built in subclass of Activity that includes a ListView without us having to create an XML layout file.

|  |  |
| --- | --- |
| INSTRUCTOR:  Create a new CallHistoryActivity.cs class file | |
| Derive from ListActivity | Explain why no layout file is required? |
| Change Label on ActivityAttribute “Call History” | Explain that this appears on the |
| Add code to read strings out of Intent and assign through ArrayAdapter<string> to ListAdapter property |  |

### Behavior in MainActivity

Now let’s add the code that stores each phone number we type, and wire-up the button to open the new history Activity we’ve created.

|  |  |
| --- | --- |
| INSTRUCTOR:  Open MainActivity.cs class file | |
| Uncomment code to locate CallHistoryButton and invoke UI for new activity | Explain Intents |
| Uncomment code to store call history #s | Explain data store |

|  |
| --- |
| INSTRUCTOR:  [RUN APP TO SHOW CHANGES – MIGHT BE USEFUL TO DEBUG AND SHOW TRANSITION IN CODE] |

### Lab Summary

In recap, you’ve seen how to add new Activity objects to your application – each Activity describing the UI and behavior for a screen. You’ve seen how to navigate to a screen and how to pass data from one screen to another using Intents, and you’ve been introduced to a second Activity type – the ListActivity which is capable of displaying lists of content.

## Thank You

Thank you for attending this \_\_\_\_\_\_\_\_ lesson – remember you can work through the labs again on your own to reinforce what we’ve covered today, and make sure to take advantage of the excellent documentation which goes into more depth on each topic we’ve covered. Let’s open up the Q&A to answer any questions you have.