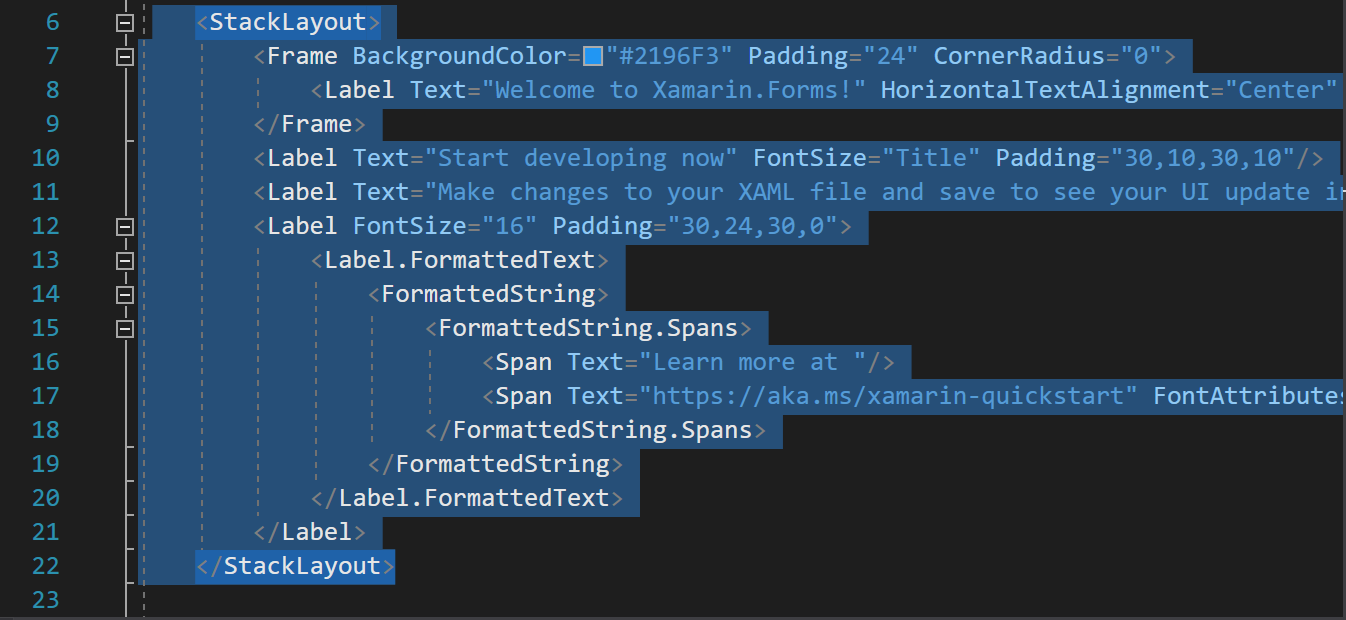
# ICE Hello RIT

Overview

Lets build our first Xamarin forms app so that we can get comfortable with using visual Studio 2019 and writing a little code.

1. Launch visual Studio 2019.
2. Choose **Create a New Project** on the right hand side of the dialog.
3. In the Search window, type in Xam and you will see a list of possible Xamarin applications that can be created. Select Xamarin Forms. Make sure to choose the version that using C# and not F#.
4. For the **Project name** type: *Hello RIT*Then click the **Create** button
5. In the **New Cross Platform App** Dialog, select **Blank**. At this point you could also turn on the Windows UWP option to create a Windows application. This is not needed for this class, but it is an option.   
     
   The click the **OK** button
6. You will notice that Visual Studio will create 3 projects (4 if you added UWP) to this solution.   
     
   Hello RIT  
   Hello RIT.Android  
   Hello RIT.iOS  
     
   On a Windows machine, the default project will be Hello RIT.Android (or Hello RIT.UWP if that was added).
7. You should now see your project in a Visual Studio window.
   * The **Solution Explorer** will have an hierarchy of the 3-4 projects in the solution.
   * To get to the project settings of the iOS or Android program, right click on that project and then go down to **Properties**.
8. To run the application on the android emulator, you will need to make sure to select the emulator and then use either F5, or the Debug menu, or the large green Play icon in the toolbar. If you have a device connected then you can select the device in the drop down menu and then use F5 to debug.
9. Let’s go and write some code to put something into the main screen. Open up the Hello RIT project and open the MainPage.xaml. This will look like the following.

  
  
Remove all of the controls (<Label> and <Frame>) from inside the <StackLayout> and add a name to the StackLayout control. Adding a name involves adding the following code to the StackLayout: x:Name="mainStack". This will allow us to use the variable named mainStack and will let us add data to this.

<StackLayout x:Name="mainStack">

</StackLayout>

Let’s go to the MainPage.xaml.cs file. You might need to hit the small triangle to the left of the MainPage.xaml. You can also get to the code by right clicking on the xaml file and going to **View Code**. Let’s go over the class definition line by line.  
  
The MainPage class is derived from the ContentPage class.  
  
The constructor is calling only the InitializeComponent(); method. This will take all of the data that is defined in the xaml and make all of the variables and nest all of the controls like it is defined in the xaml.  
  
This means that the mainStack variable will be available **AFTER** this is called and not before.

1. Let’s add some code to display some things on the screen. Right after the InitializeComponent();, we will be making 2 variables and getting them to display.  
     
    var topLabel = new Label();  
    var bottomLabel = new Label();
2. Now add the following code after the variable declaration inside the constructor:  
     
    topLabel.Text = "Hello";  
    topLabel.FontSize = 36;  
    mainStack.Children.Add(topLabel);
3. Run the program. You should see the following:

A picture containing screenshot

Description automatically generated  
The last line is adding the label to the StackLayout named mainStack.

1. There are multiple ways to move the label to a new position. You can add padding, margins (like html) or translate the control. Add the following line:  
     
    topLabel.Margin = new Thickness(100, 40, 0, 0);
2. Now go ahead and set properties of the bottomLabel variable:

**(you write this code!)**

* **Give it text of RIT**
* **Give it a font size of 48**
* **Center the text using** HorizontalTextAlignment
* **Add it to the mainStack**

**Lastly, let’s give the entire page an orange color with this line of code** BackgroundColor = Color.Orange;

1. It should look something like this:  
     
   A screenshot of a cell phone

   Description automatically generated
2. Lastly, add your name somewhere on the screen by creating a new label and then configuring its properties.

A screenshot of a cell phone

Description automatically generated

*Optional Challenge: Can you determine the height and width of the screen (programmatically) and then place the name label 50 points up from the bottom of the screen and centered. (Hint: look at VerticalOptions)*

*Also, give this new label a grayish color.*

**Submission: ZIP and Post to the dropbox before the due date**

**Discussion:**

* Here we experimented with some of the basic UI components in a single Page application. A typical mobile app has multiple pages filled with tables, buttons, and other customized views.
* Check out the **App.xaml.cs** file. This file manages the “application lifecycle” (the app starting, pausing, resuming, etc). This is a good place to load use data when the app starts up and save state whenever the use hits the home button or gets a phone call.