+ Queens University Belfast

Android Studio

CSC3054 / CSC7054

Event Build App

Exercise 1 – Creating the Event Build app

In this app, create three Views. A `TextView` (to hold some text) and two `Button` components (these will complete a task when clicked).

Before You Begin

Open `Android Studio` and create a new project called "`EventBuild`". Refer to the '`Creating your first project`' tutorial to help you create a project. Once created your project should look like figure 1. Switch from `Design` view to Text `view` .

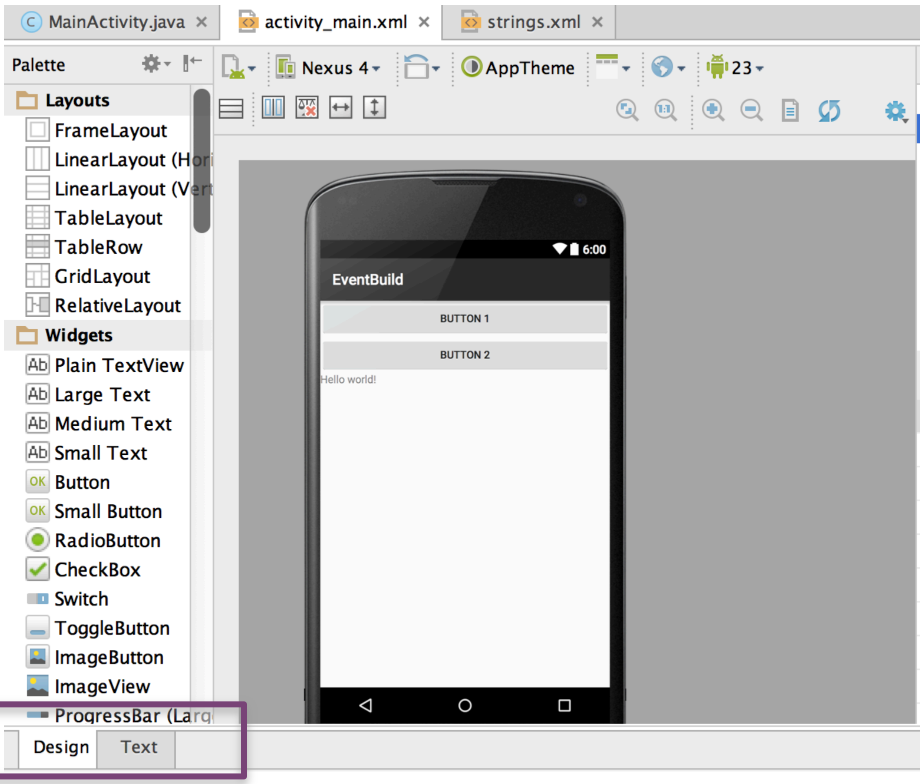


FIGURE 1 - OPEN PROJECT

Step 1 - Create the View components

Once in the `Text view` of the `activity_main.xml` file, add the following components and attributes:

Component	Attribute	Value	Notes
`LinearLayout`	`android:layout_width`	`="fill_parent"`	
	`android:layout_height`	`="fill_parent"`	
	`android:orientation`	`="vertical"`	
`Button`	`android:id`	`="@+id/button_1"`	Remember to add an `id` for the `view` and create a string resource
	`android:layout_height`	`="wrap_content"`	
	`android:layout_width`	`="match_parent"`	
	`android:text`	`="@string/button_1"`	
`Button`	`android:id`	`="@+id/button_2"`	Remember to add an `id` for the `view` and create a string resource
	`android:layout_height`	`="wrap_content"`	
	`android:layout_width`	`="match_parent"`	
	`android:text`	`="@string/button_2"`	
`TextView`	`android:id`	`="@+id/text_id"`	Remember to add an `id` for the `view` and create a string resource
	`android:layout_width`	`="wrap_content"`	
	`android:layout_height`	`="wrap_content"`	
	`android:capitalize`	`="characters"`	
	`android:text`	`="@string/hello_world"`	

When rendered, the application should look like figure 2.

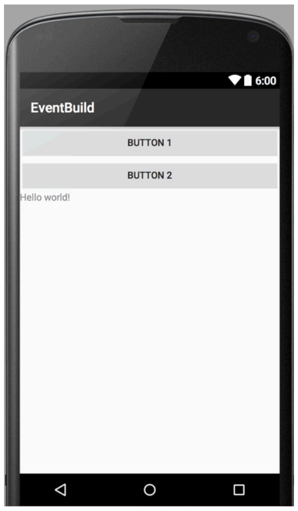


Figure 2 - Design view of app

Step 2 - Check the `strings.xml` file

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">EventBuild</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="button_1">Button 1</string>
    <string name="button_2">Button 2</string>

</resources>
```

Step 3 – Add the functionality

Open the `MainActivity.java` file from the project explorer menu as shown in figure 3.

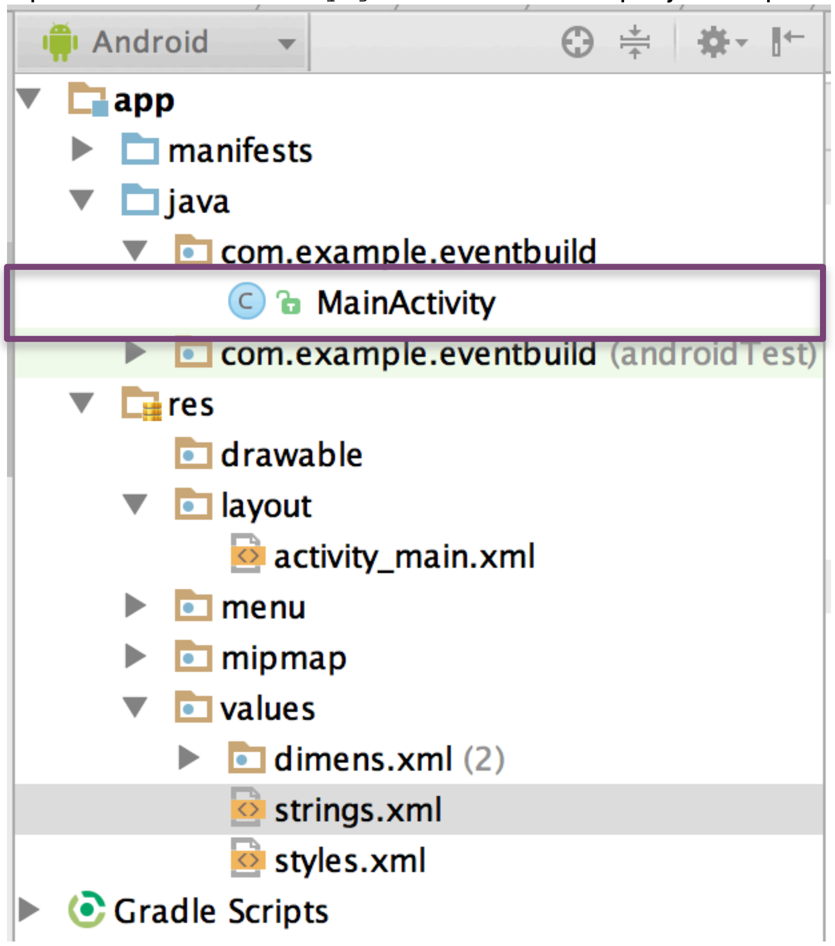


FIGURE 3- MAINACTIVITY.JAVA

Task 1 - Ensure that `MainActivity` inherits from `Activity`

The `Activity` class is an object dedicated to a specific screen of an app. Think of it as something that handles and displays all the associated work that drives the functionality for that screen.

Open `MainActivity.java` and ensure that this java file extends Activity and imports the `android.app.Activity` library.

```
package com.example.eventbuild;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {

}
```

The `Activity` doesn't yet know its purpose because you haven't told it what to do or how to appear. You can create your activity's appearance, aka `Layout`, one of two ways:

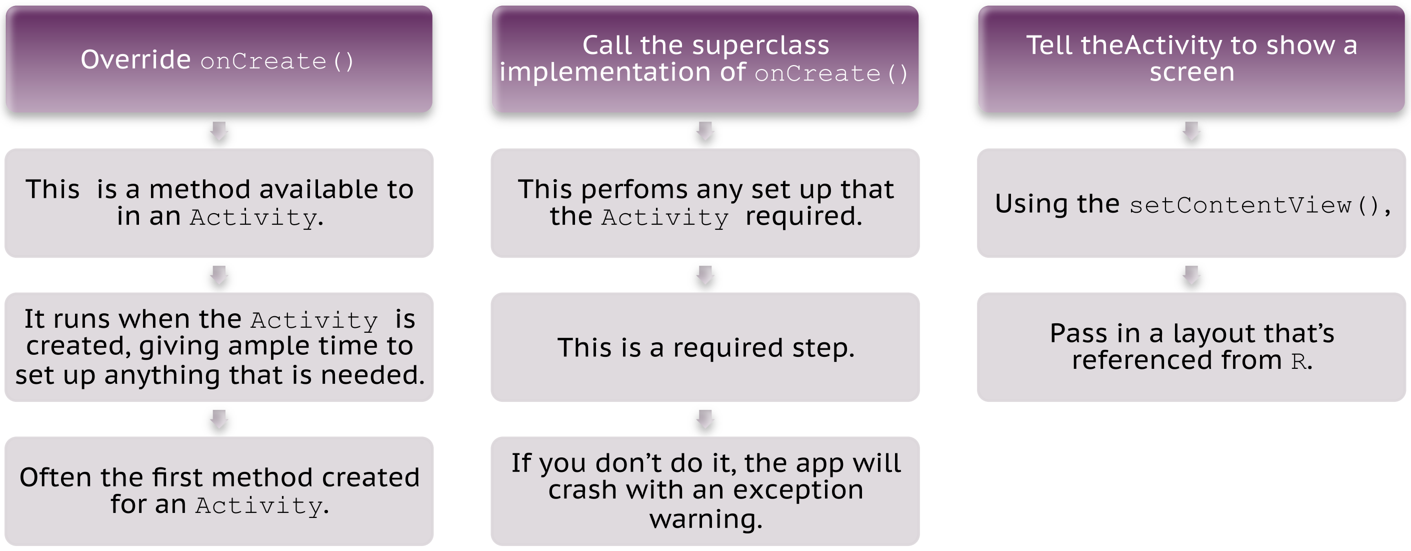
- Declare it in an `XML file`
- Programmatically create it in a `Java class`

The more common approach is creating the layout in an `XML file` (which we completed in the previous step). You'd take the programmatic approach if you need to tailor some aspect of your layout.

The following code snippet demonstrates how you would specify that an activity should determine its appearance using the XML layout in `res/layout/activity_main_menu.xml`.

```
public class MainActivity extends Activity {

   @Override

   protected void onCreate(Bundle savedInstanceState) {

     super.onCreate(savedInstanceState);

     setContentView(R.layout.activity_main);

  }

}
```

What the code does



Override `onCreate()`

This is a method available to in an `Activity`.

It runs when the `Activity` is created, giving ample time to set up anything that is needed.

Often the first method created for an `Activity`.

Call the superclass implementation of `onCreate()`

This perfoms any set up that the `Activity` required.

This is a required step.

If you don't do it, the app will crash with an exception warning.

Tell theActivity to show a screen

Using the `setContentView()`,

Pass in a layout that's referenced from `R`.

The R File

This is a special class that is generated by Android for an app's various assets. These could be a screen layout like you see above, or it could be something like a localized string, image or animation. Android generates the `R` file while building, so you shouldn't modify it at all.

Task 2- Access the views in the UI

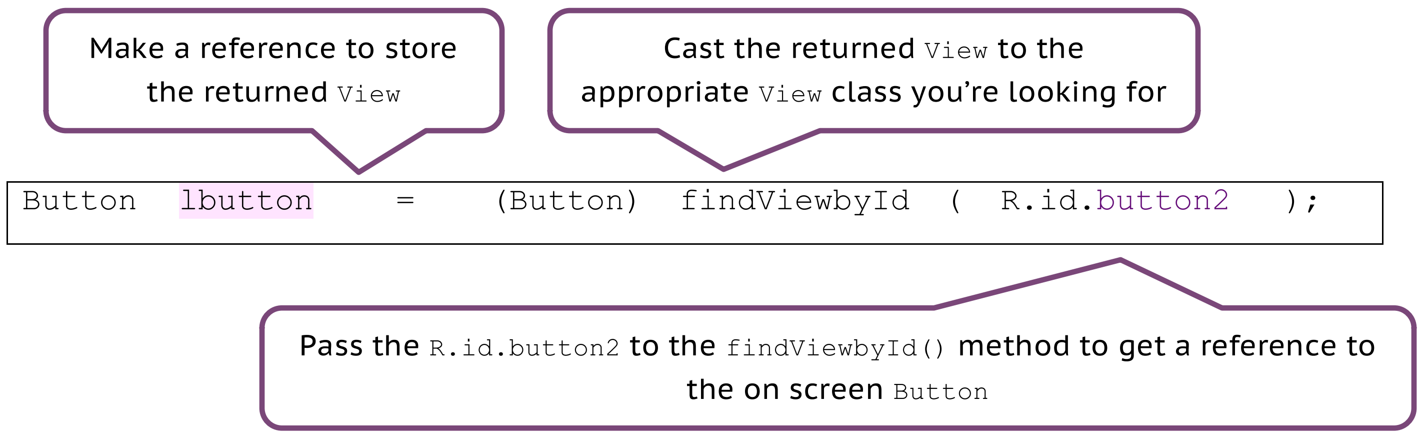
The XML layout has a number of view components that require functionality. In order to access these through the `R` File, they must first be declared and initialized.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //--- find both the buttons---
        Button sButton = (Button) findViewById(R.id.button_1);
        Button lButton = (Button) findViewById(R.id.button_2);
    }
```

Getting view references

You need to assign references for each of your app resources (`Button, TextView` etc). Android provides a utility method called `findViewById()` to do just that. The `findViewbyId()` method is the base class of the `Activity`, so it can be used directly in the `MainActivity` class since it is a subclass of `Activity`. This method searches the XML layout for view objects specified by their ID, allowing you to manipulate them from Java.

The `findViewById()` method takes on parameter, the `R` constant for the `View`. However, since the method is mean to be generic, it returns a `View` not one of the `View` subclasses (like `Button`, `TextView`, or any other `View`). It's easy enough though, you just need to cast the result to the `View` you're expecting.


Make a reference to store the returned View
Cast the returned View to the appropriate View class you're looking for
Pass the R.id.button2 to the findViewbyId() method to get a reference to the on screen Button

```
Button  lbutton    =    (Button)  findViewbyId  (  R.id.button2   );
```

Task 3 - Interact with the view components

Once you have access to the individual view components within the Java code functionality can be added. The following code segment adds an `onClick()` action to each button. When the user clicks a button the text will either get bigger or smaller.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // -- register click event with first button ---
        sButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // --- find the text view --
                TextView txtView = (TextView) findViewById(R.id.text_id);
                // -- change text size --
                txtView.setTextSize(14);
            }
        });
        // -- register click event with second button ---
        lButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // --- find the text view --
                TextView txtView = (TextView) findViewById(R.id.text_id);
                // -- change text size --
                txtView.setTextSize(24);
            }
        });

    }
```

The code above runs the `onClick()` method every time either button is clicked. In the `onClick()` method for each button, the `TextView` is declared and initialized. When the user clicks on either button, the text size of the string contained within the `TextView` will change (using the `setTextSize()` method) to either size `14` or `24` respectively.

By using this simple approach to link views to a class and provide actions to perform on a certain event, highly complex activities can be built.

Full Code for `MainActivity.java`

```java
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //--- find both the buttons---
        Button sButton = (Button) findViewById(R.id.button_1);
        Button lButton = (Button) findViewById(R.id.button_2);

        // -- register click event with first button ---
        sButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // --- find the text view --
                TextView txtView = (TextView) findViewById(R.id.text_id);
                // -- change text size --
                txtView.setTextSize(14);
            }
        });

        // -- register click event with second button ---
        lButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // --- find the text view --
                TextView txtView = (TextView) findViewById(R.id.text_id);
                // -- change text size --
                txtView.setTextSize(24);
            }
        });

    }
}
```

Step 4 – Test the app in the Emulator

Press the play button and run the app in the emulator as shown in figure 4.

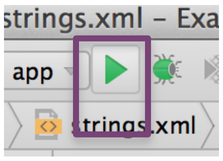


FIGURE 4 - RUNNING AN APP

When it runs in the emulator it should look like figure 5.

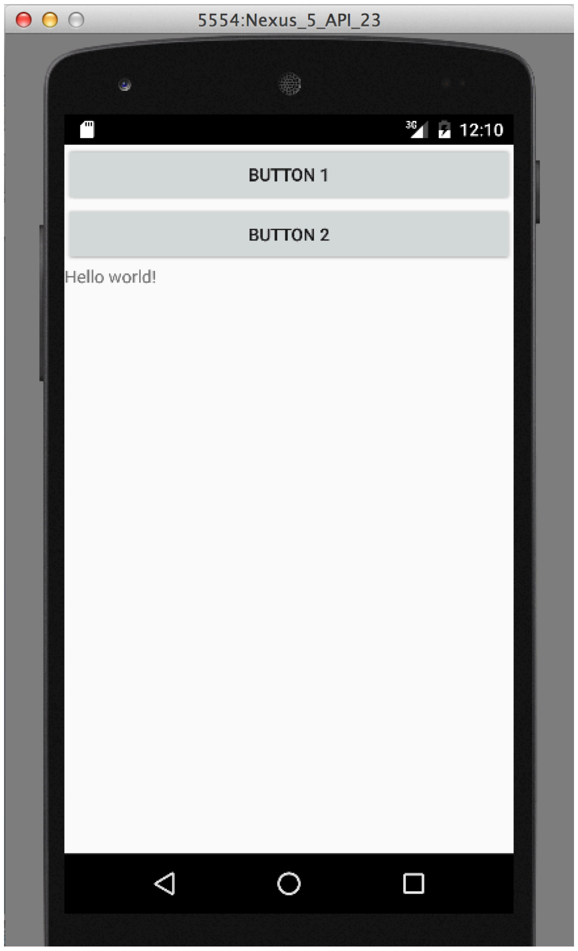


FIGURE 5 - COMPLETED APP

Test that all functionality works.