# Connect Java with XML widgets

To make your project interactive, you will build code, in Java. Java will be used to tell your app what to do, when the user interacts with widgets, or other components, in your game. In our case, we have a button widget, that we need to hook up to do something, when the user clicks it. To connect the widget to Java, we need to use the IDs, and create objects, in the Java code, to link to the widget IDs, in the activity. Let's start with a content\_main.xml file located in the project panel, under res, layout, content\_main.xml.

If we look at the xml code, in text view, we will see our two widgets, our text view, and button widgets, we worked with earlier. Our text view displays text, but, we don't have an ID name associated with it. In order for our program to access the widget, in code, we need to give it an ID. We're going to wire our app, so that when we click the button, it displays a message, in the text view widget. So, the first thing we need to do, is to give our widget a name. Along the way, we'll also resolve the error we see in our button widget, below. In our code, add a new line to add in a new attribute, in our text view widget.

To create an ID, we will use the android:id attribute name. And, we will set that equal to a blank string. To set the ID name, we need to use a special string of text, to register the ID correctly. Inside of the quotation marks, for the property, start with the @ symbol, then +, then enter id, and, then, a forward slash. In essence, we are adding a new ID, to our project. Then, you need to provide the name you want to call the widget. This text view will display the result of our roll, in the game.

So, let's call it rollResult. There, our text view has an ID, now. If we switch back to design view, select the text view widget, and look at the properties, you will see that the ID name is there, as well. Let's switch back to the code. In our activity, we are using the relative layout widget, to arrange our layout. Part of how this works, is to say where widgets are, in relationship to other widgets. We have an error in our button widget, because we are saying that it is below a widget, called text view. But, the problem is, is that we don't have anything with the ID of text view, so, that doesn't exist, and, we get our error.

We can fix that, by telling our button widget to layout below our text view, with the new ID we just created, rollResult. Update the attribute, to use the new ID name. Under android, under layout below, change textView, so that now it says rollResult. Now, the button has an object, to relate its position to. And, both widgets have IDs we can use, in our Java code. The next thing we need to do, is to tell the button that we want to run a Java method, when the user clicks it.

We'll use a new attribute, called on click, and provide it with a name of a Java method that we'll create, in a little bit. In the code, create a new line, in the button tag, to create a new attribute. Access the android:onClick attribute. Next, we will add in the method name that we'll create, in the next step, called rollDice. You may get an error, when you type that in, because the method doesn't exist, yet, but, we'll fix that. Also note that we don't need a pair of parentheses, when we declare the method name.

Now, open the main activity.java document. If it isn't open, go the project panel, and open up the java folder, and, then, open the com.example.diceout folder, and, then, double click MainActivity. To create a link between the widget, in XML, and the code, in Java, we need to create an instance of the widget class, and attach it to an item, in the activity. We want to scope this to the entire activity, so, we will define this instance, at the top of our class. At the top, above our score variable, that we created earlier, we'll create two things.

First, create a text view, called rollResults. When you type in text view, it will appear red, that is because the class hasn't been added, to our project. You'll see a notification to press the Alt, and Enter keys. This will add an import statement, at the top of our code, to include this class, with our code. Next, create a new button, called rollButton. Import the class, again, using the Alt Enter shortcut. That's half of what we have to do. The next, is to access each of these, from the activity view, and assign them, to their instance names.

We need to do this, when the layout is created. Luckily, we have a method for us to do exactly that, called onCreate. At the end of that method, we need to add two lines of code. First, for the text view, we will use the rollResult instance, that we created, and assign that, to the widget. To access the widget, we will use the method findViewById. This method will find the widget, in the activity layout, and assign it to the instance, for us to use, in the code. Inside of the method, let's start with rollResult, and we'll set that equal to the findViewById method.

Inside of the method, we need to provide the ID. We start with R, which refers to the project, and, then, id, and, then, we enter in the name of the ID we want to access, in this case, rollResult. But, you see that we'll get an error. If you highlight it, with your mouse, you'll see that we have an incompatible type error. We have a type mismatch, and, we need to use a cast, which will convert the view that we get, for the method, to a compatible text view object, to assign to the instance. Add in a cast, to do that, before the method call.

Now, we need to repeat this, for the button, rollButton, equals cast, as a button, findViewById, R.id.rollButton. And that's it, we have now linked an instance, in Java, to the widget, in our activity layout. Now, we need to create the method that we will call, when we click the button widget. We could add this below the onCreate method. We'll create a new public method, called rollDice.

When the on click event happens, it will pass in the view of the event, that we need to accept, in the method, we'll just call it v. Now, we can access our text view widget, and send a message. We will access the rollResult instance, and, then, use a method called setText, which is part of the text view class. We'll then put in the text we want to add. We'll just put a simple message called Clicked. With that, let's go ahead, and test, and see what happens. Open up the Run menu, and select Run app. Select our virtual device, using OK.

Click the button with the mouse, and, you'll see, our custom message will appear, in the text view widget. Linking Java, and activity layouts, from XML, requires two main steps. In your XML, you need to define an ID, for each object you want to access, in the code. Then, in Java, you create instances, of the widget classes, and, then, assign them views, using the findViewById method, casting, as needed, based on the widget types. Then, you can access methods of the widgets, and build your app.