56. An Android TableLayout and TableRow Tutorial

When the work began on the next chapter of this book (*"An Android SQLite Database Tutorial"*) it was originally intended that it would include the steps to design the user interface layout for the database example application. It quickly became evident, however, that the best way to implement the user interface was to make use of the Android TableLayout and TableRow views and that this topic area deserved a self-contained chapter. As a result, this chapter will focus solely on the user interface design of the database application completed in the next chapter, and in doing so, take some time to introduce the basic concepts of table layouts in Android Studio.

56.1 The TableLayout and TableRow Layout Views

The purpose of the TableLayout container view is to allow user interface elements to be organized on the screen in a table format consisting of rows and columns. Each row within a TableLayout is occupied by a TableRow instance, which, in turn, is divided into cells, with each cell containing a single child view (which may itself be a container with multiple view children).

The number of columns in a table is dictated by the row with the most columns and, by default, the width of each column is defined by the widest cell in that column. Columns may be configured to be shrinkable or stretchable (or both) such that they change in size relative to the parent TableLayout. In addition, a single cell may be configured to span multiple columns.

Consider the user interface layout shown in Figure 56-1:

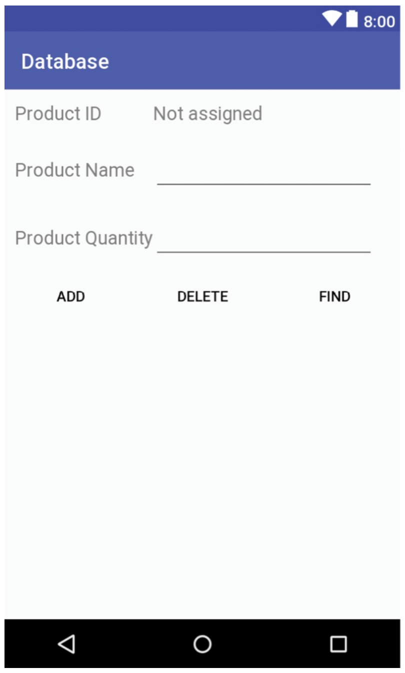


Figure 56-1

From the visual appearance of the layout, it is difficult to identify the TableLayout structure used to design the interface. The hierarchical tree illustrated in Figure 56-2, however, makes the structure a little easier to understand:

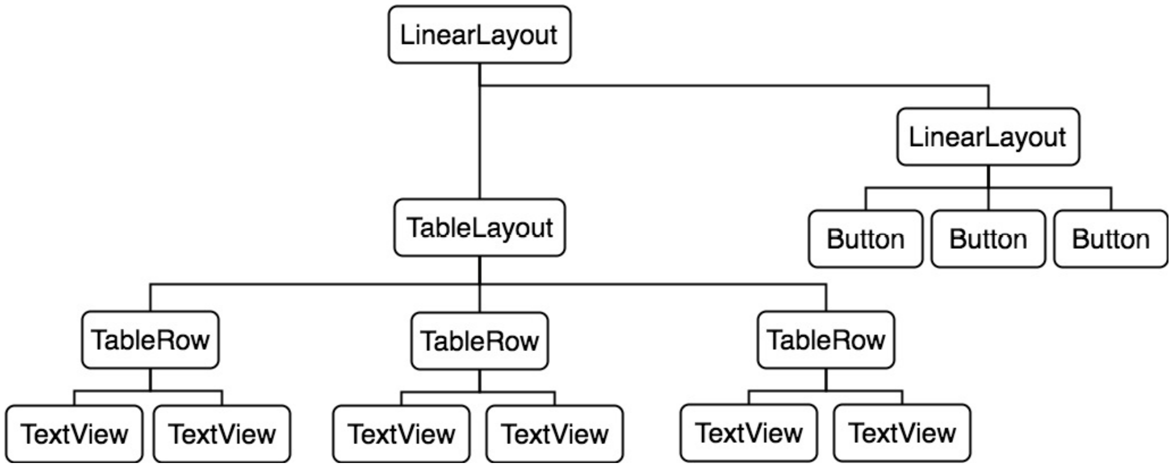


Figure 56-2

Clearly, the layout consists of a parent LinearLayout view with TableLayout and LinearLayout children. The TableLayout contains three TableRow children representing three rows in the table. The TableRows contain two child views, with each child representing the contents of a column cell. The LinearLayout child view contains three Button children.

The layout shown in Figure 56-2 is the exact layout that is required for the database example that will be completed in the next chapter. The remainder of this chapter, therefore, will be used to work step by step through the design of this user interface using the Android Studio Layout Editor tool.

56.2 Creating the Database Project

Start Android Studio and create a new project, entering *Database* into the Application name field and *ebookfrenzy.com* as the Company Domain setting before clicking on the *Next* button.

On the form factors screen, enable the *Phone and Tablet* option and set the minimum SDK setting to API 14: Android 4.0 (IceCreamSandwich). Continue to proceed through the screens, requesting the creation of an Empty Activity named *DatabaseActivity* with a corresponding layout file named *activity_database.*

56.3 Adding the TableLayout to the User Interface

Locate the *activity_database.xml* file in the Project tool window (*app* -> *res* -> *layout*) and double-click on it to load it into the Layout Editor tool. By default, Android Studio has used a ConstraintLayout as the root layout element in the user interface. This needs to be replaced by a vertically oriented LinearLayout. With the Layout Editor tool in Text mode, replace the XML with the following:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
</LinearLayout>
```

Switch to Design mode and, referring to the Layouts category of the Palette, drag and drop a TableLayout view so that it is positioned at the top of the LinearLayout canvas area. With the LinearLayout component selected, use the Attributes tool window to set the layout_height property to *wrap_content*.

Once these initial steps are complete, the Component Tree for the layout should resemble that shown in Figure 56-3.

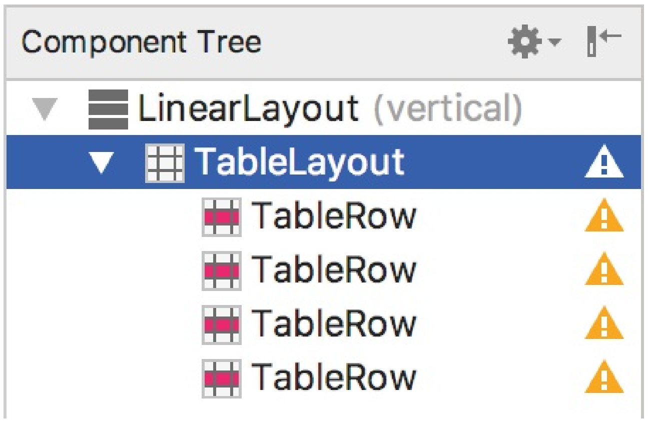


Figure 56-3

Clearly, Android Studio has automatically added four TableRow instances to the TableLayout. Since only three rows are required for this example, select and delete the fourth TableRow instance. Additional rows may be added to the TableLayout at any time by dragging the TableRow object from the palette and dropping it onto the TableLayout entry in the Component Tree tool window.

56.4 Configuring the TableRows

From within the *Text* section of the palette, drag and drop two TextView objects onto the uppermost TableRow entry in the Component Tree (Figure 56-4):

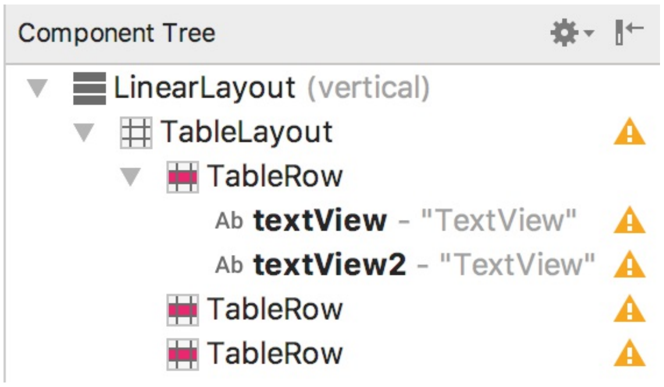


Figure 56-4

Select the left most TextView within the screen layout and, in the Attributes tool window, change the *text* property to “Product ID”. Repeat this step for the right most TextView, this time changing the text to “Not assigned” and specifying an *ID* value of *productID*.

Drag and drop another TextView widget onto the second TableRow entry in the Component Tree and change the text on the view to read "Product Name". Locate the Plain Text object in the palette and drag and drop it so that it is positioned beneath the Product Name TextView within the Component Tree as outlined in Figure 56-5. With the TextView selected, change the inputType property from textPersonName to None, delete the "Name" string from the text property and set the ID to *productName*.

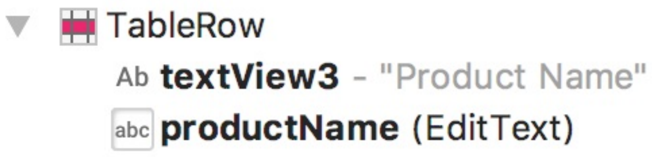

TableRow
Ab textView3 - "Product Name"
abc productName (EditText)

Figure 56-5

Drag and drop another TextView and a Number (Decimal) Text Field onto the third TableRow so that the TextView is positioned above the Text Field in the hierarchy. Change the text on the TextView to *Product Quantity* and the ID of the Text Field object to *productQuantity*.

Shift-click to select all of the widgets in the layout as shown in Figure 56-6 below, and use the Attributes tool window to set the textSize property on all of the objects to 18sp:

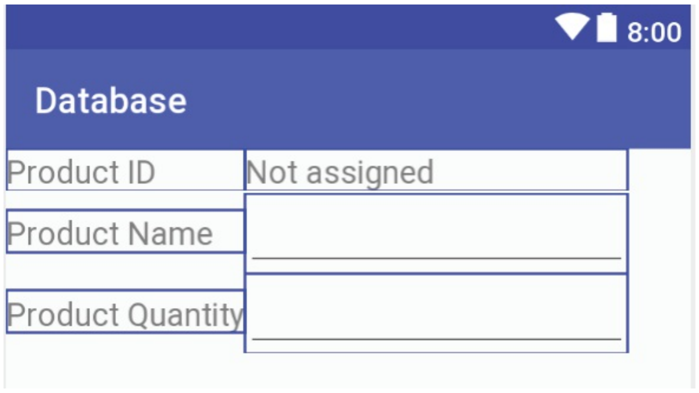


Figure 56-6

Before proceeding, be sure to extract all of the text properties added in the above steps to string resources.

56.5 Adding the Button Bar to the Layout

The next step is to add a LinearLayout (Horizontal) view to the parent LinearLayout view, positioned immediately below the TableLayout view. Begin by clicking on the small arrow to the left of the TableLayout entry in the Component Tree so that the TableRows are folded away from view. Drag a *LinearLayout (Horizontal)* instance from the *Layouts* section of the Layout Editor palette, drop it immediately beneath the TableLayout entry in the Component Tree panel and change the layout_height property to *wrap_content*:

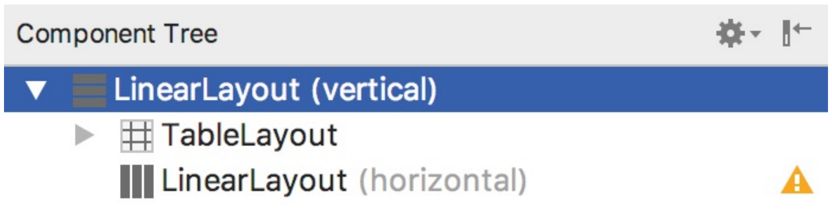
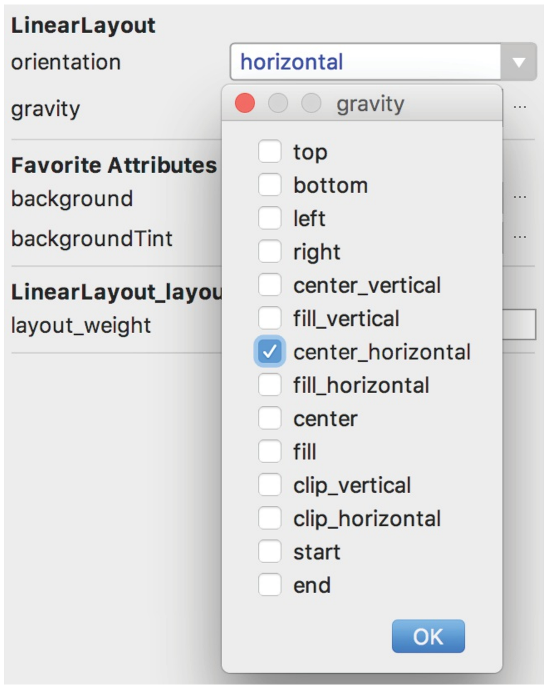


Figure 56-7

Drag and drop three Button objects onto the new LinearLayout and assign string resources for each button that read “Add”, “Find” and “Delete” respectively. Buttons in this type of button bar arrangement should generally be displayed with a borderless style. For each button, use the Attributes tool window to change the style setting to *Widget.AppCompat.Button.Borderless.*

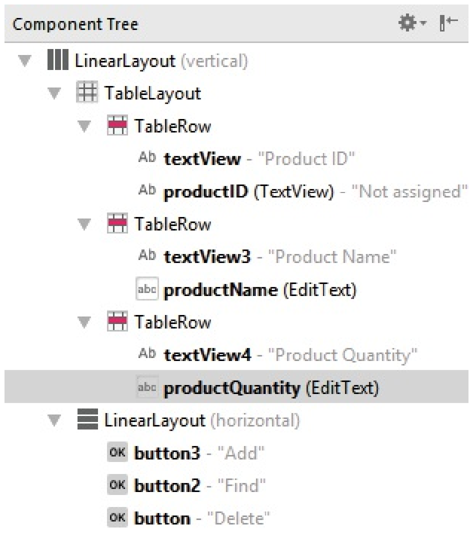
With the new horizontal Linear Layout view selected in the Component Tree change the gravity property to *center_horizontal* (Figure 56-8) so that the buttons are centered horizontally within the display:


LinearLayout
orientation: horizontal
gravity

- [ ] top
- [ ] bottom
- [ ] left
- [ ] right
- [ ] center_vertical
- [ ] fill_vertical
- [✓] center_horizontal
- [ ] fill_horizontal
- [ ] center
- [ ] fill
- [ ] clip_vertical
- [ ] clip_horizontal
- [ ] start
- [ ] end

Figure 56-8

Before proceeding, check the hierarchy of the layout in the Component Tree panel, taking extra care to ensure the view ID names match those in the following figure:

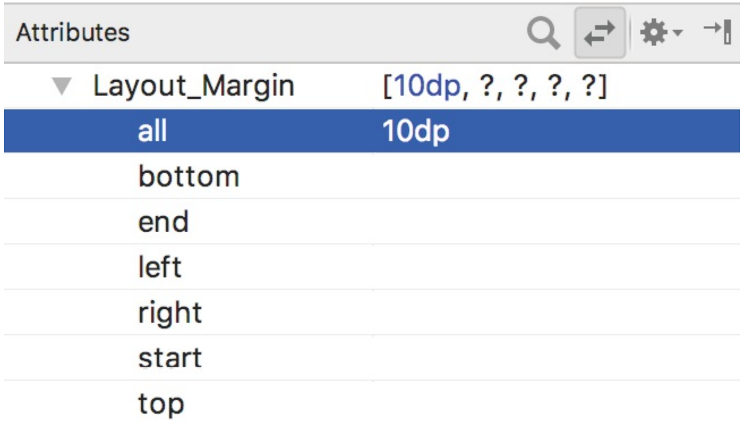


```
Component Tree
▼ LinearLayout (vertical)
    ▼ TableLayout
        ▼ TableRow
            textView - "Product ID"
            productID (TextView) - "Not assigned"
        ▼ TableRow
            textView3 - "Product Name"
            productName (EditText)
        ▼ TableRow
            textView4 - "Product Quantity"
            productQuantity (EditText)
    ▼ LinearLayout (horizontal)
        button3 - "Add"
        button2 - "Find"
        button - "Delete"
```

Figure 56-9

56.6 Adjusting the Layout Margins

All that remains is to adjust some of the layout settings. Begin by clicking on the first TableRow entry in the Component Tree panel so that it is selected. Hold down the Cmd/Ctrl-key on the keyboard and click in the second and third TableRows so that all three items are selected. In the Attributes panel, list all attributes, locate the *layout_margins* attributes category and, once located, change all the *all* value to 10dp as shown in Figure 56-10:



Attributes	
▼ Layout_Margin	[10dp, ?, ?, ?, ?]
all	10dp
bottom	
end	
left	
right	
start	
top	

Figure 56-
10

With margins set on all three TableRows, the user interface should appear as illustrated in Figure 56-1.

56.7 Summary

The Android TableLayout container view provides a way to arrange view components in a row and column configuration. While the TableLayout view provides the overall container, each row and the cells contained therein are implemented via instances of the TableRow view. In this chapter, a user interface has been designed in Android Studio using the TableLayout and TableRow containers. The next chapter will add the functionality behind this user interface to implement the SQLite database capabilities.