Android Development

02 Practical (b) The layout editor

# Overview

You just built a simple App using a ConstraintLayout.

ConstraintLayout was designed to make it easy to drag UI elements into the layout editor.

ConstraintLayout is a [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html), which is a special View that can contain other View objects (called *children* or *child views*). This practical shows more features of ConstraintLayout and the layout editor.

This practical also introduces two other [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) subclasses:

* LinearLayout: A group that aligns child View elements within it horizontally or vertically.
* RelativeLayout: A group of child View elements in which each View element is positioned and aligned relative to other View element within the ViewGroup. Positions of the child View elements are described in relation to each other or to the parent ViewGroup.

## What you'll do

* Create a layout variant for a horizontal display orientation.
* Create a layout variant for tablets and larger displays.
* Modify the layout to add constraints to the UI elements.
* Use ConstraintLayout baseline constraints to align elements with text.
* Use ConstraintLayout pack and align buttons to align elements.

## App Overview

The Hello Toast app in a previous practical uses ConstraintLayout to arrange the UI elements in the Activity layout, as shown in the figure below.

A screenshot of a cell phone

Description automatically generated

To gain more practice with ConstraintLayout, you will create a variant of this layout for horizontal orientation as shown in the figure below.

A screenshot of a cell phone

Description automatically generated

You will also learn how to use baseline constraints and some of the alignment features of ConstraintLayout by creating another layout variant for tablet displays.

A screenshot of a cell phone

Description automatically generated

You also play with other ViewGroup subclasses such as [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) and [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html), and change the Hello Toast app layout to use them.

# Create Layout Variants

In this task you will create variants of your layout for horizontal (also known as *landscape*) and vertical (also known as *portrait*) orientations for phones, and for larger displays such as tablets.

You will be using some of the buttons in the top two toolbars of the layout editor. The top toolbar lets you configure the appearance of the layout preview in the layout editor:

A close up of a sign

Description automatically generated

In the figure above:

1. **Select Design Surface**: Select **Design** to display a color preview of your layout, or **Blueprint** to show only outlines for each UI element. To see *both* panes side by side, select **Design + Blueprint**.
2. **Orientation in Editor**: Select **Portrait** or **Landscape** to show the preview in a vertical or horizontal orientation. This is useful for previewing the layout without having to run the app on an emulator or device. To create alternative layouts, select **Create Landscape Variation** or other variations.
3. **Device in Editor**: Select the device type (phone/tablet, Android TV, or Android Wear).
4. **API Version in Editor**: Select the version of Android to use to show the preview.
5. **Theme in Editor**: Select a theme (such as **AppTheme**) to apply to the preview.
6. **Locale in Editor**: Select the language and locale for the preview. This list displays only the languages available in the string resources (see the lesson on localization for details on how to add languages). You can also choose **Preview as Right To Left** to view the layout as if an RTL language had been chosen.

The second toolbar lets you configure the appearance of UI elements in a ConstraintLayout, and to zoom and pan the preview:

A close up of a logo

Description automatically generated

In the figure above:

1. **Show**: Choose **Show Constraints** and **Show Margins** to show them in the preview, or to stop showing them.
2. **Autoconnect**: Enable or disable Autoconnect. With Autoconnect enabled, you can drag any element (such as a Button) to any part of a layout to generate constraints against the parent layout.
3. **Clear All Constraints**: Clear all constraints in the entire layout.
4. **Infer Constraints**: Create constraints by inference.
5. **Default Margins**: Set the default margins.
6. **Pack**: Pack or expand the selected elements.
7. **Align**: Align the selected elements.
8. **Guidelines**: Add vertical or horizontal guidelines.
9. Zoom/pan controls: Zoom in or out.

## **Create a layout variant for horizontal orientation**

Use the **Orientation in Editor** button  to compare the layout in landscape and portrait mode.

The visual difference between vertical and horizontal orientations for this layout is that the digit (0) in the show\_count TextView element is too low for the horizontal orientation—too close to the **Count** button. Depending on which device or emulator you use, the TextView element may appear too large or not centred because the text size is fixed to 160sp.

To fix this for horizontal orientations while leaving vertical orientations alone, you can create variant of the Hello Toast app layout that is different for a horizontal orientation. Follow these steps:

1. Click the **Orientation in Editor** button  in the top toolbar.
2. Choose **Create Landscape Variation**.

A new editor window opens with the **land/activity\_main.xml** tab showing the layout for the landscape (horizontal) orientation. You can change this layout, which is specifically for horizontal orientation, without changing the original portrait (vertical) orientation.

1. In the **Project > Android** pane, look inside the **res > layout** directory, and you will see that Android Studio automatically created the variant for you, called activity\_main.xml (land).

A screenshot of a cell phone

Description automatically generated

## Change the layout for horizontal orientation

You can use the Attributes pane in the **Design** tab to set or change attributes, but it can sometimes be quicker to use the **Text** tab to edit the XML code directly.

To change the layout, follow these steps:

1. The **land/activity\_main.xml** tab should still be open in the layout editor; if not, double-click the **activity\_main.xml (land)** file in the **layout** directory.
2. Click the **Text** tab and the **Preview** tab (if not already selected).
3. Find the TextView element in the XML code.
4. Change the android:textSize="160sp" attribute to android:textSize="120sp". The layout preview shows the result:
5. Choose different devices in the **Device in Editor** dropdown menu to see how the layout looks on different devices in horizontal orientation.

In the editor pane, the **land/activity\_main.xml** tab shows the layout for horizontal orientation. The **activity\_main.xml** tab shows the unchanged layout for vertical orientation. You can switch back and forth by clicking the tabs.

1. Run the app on an emulator or device and switch the orientation from vertical to horizontal to see both layouts.

## **Create a layout variant for tablets**

You can preview the layout for different devices by clicking the **Device in Editor** button  in the top toolbar. If you pick a device such as **Nexus 10** (a tablet) from the menu, you can see that the layout is not ideal for a tablet screen—the text of each Button is too small, and the arrangement of the Button elements at the top and bottom is not ideal for a large-screen tablet.

To fix this for tablets while leaving the phone-size horizontal and vertical orientations alone, you can create variant of the layout that is completely different for tablets. Follow these steps:

1. Click the **Design** tab (if not already selected) to show the design and blueprint panes.
2. Click the **Orientation in Editor** button  in the top toolbar.
3. Choose **Create layout x-large Variation**.

A new editor window opens with the **xlarge/activity\_main.xml** tab showing the layout for a tablet-sized device. The editor also picks a tablet device, such as the Nexus 9 or Nexus 10, for the preview. You can change this layout, which is specifically for tablets, without changing the other layouts.

## Change the layout for Tablets

You can use the Attributes pane in the **Design** tab to change attributes for this layout.

1. Turn off the Autoconnect tool in the toolbar. For this step, ensure that the tool is disabled: 
2. Clear all constraints in the layout by clicking the **Clear All Constraints** button in the toolbar.

With constraints removed, you can move and resize the elements on the layout freely.

1. The layout editor offers resizing handles on all four corners of an element to resize it. In the **Component Tree**, select the TextView called show\_count. To get the TextView out of the way so that you can freely drag the Button elements, drag a corner of it to resize it.

Resizing an element hardcodes the width and height dimensions. Avoid hardcoding the size dimensions for most elements, because you can't predict how hardcoded dimensions will look on screens of different sizes and densities. You are doing this now just to move the element out of the way, and you will change the dimensions in another step.

1. Select the button\_toast Button in the **Component Tree**, click the **Attributes** tab to open the **Attributes** pane, and change the textSize to **60sp** (#1 in the figure below) and the layout\_width to wrap\_content (#2 in the figure below).

A screenshot of a cell phone

Description automatically generated

As shown on the right side of the figure above (2), you can click the view inspector's width control, which appears in two segments on the left and right sides of the square, until it shows Wrap Content. As an alternative, you can select **wrap\_**content from the layout\_width menu.

You use wrap\_content so that if the Button text is localized into a different language, the Button will appear wider or thinner to accommodate the word in the different language.

1. Select the button\_count Button in the Component Tree, change the textSize to 60sp and the layout\_width to wrap\_content, and drag the Button above the TextView to an empty space in the layout.

## **Use a baseline constraint**

You can align one UI element that contains text, such as a TextView or Button, with another UI element that contains text. A *baseline constraint* lets you constrain the elements so that the text baselines match.

1. Constrain the button\_toast Button to the top and left side of the layout, drag the button\_count Button to a space near the button\_toast Button, and constrain the button\_count Button to the left side of the button\_toast Button.
2. Using a *baseline constraint*, you can constrain the button\_count Button so that its text baseline matches the text baseline of the button\_toast Button. Select the button\_count element, and then hover your pointer over the element until the baseline constraint button  appears underneath the element. (you may have to right click the element and select ‘Show Baseline’ – depends on the exact IDE installation)
3. Click the baseline constraint button. The baseline handle appears, blinking in green. Click and drag a baseline constraint line to the baseline of the button\_toast element.

## **Expand the buttons horizontally**

The pack button  in the toolbar provides options for packing or expanding selected UI elements. You can use it to equally arrange the Button elements horizontally across the layout.

1. Select the button\_count Button in the **Component Tree**, and Shift-select the button\_toast Button so that both are selected.
2. Click the pack button  in the toolbar, and choose **Expand Horizontally** as shown in the figure below.

The Button elements expand horizontally to fill the layout as shown below.

A screenshot of a cell phone

Description automatically generated

1. To finish the layout, constraint the show\_count TextView to the bottom of the button\_toast Button and to the sides and bottom of the layout.
2. The final steps are to change the show\_count TextView layout\_width and layout\_height to Match **Constraints** and the textSize to 200sp. The final layout looks like the figure below.

A screenshot of a cell phone

Description automatically generated

1. Click the **Orientation in Editor** button  in the top toolbar and choose **Switch to Landscape**.
2. Run the app on different emulators, and change the orientation after running the app, to see how it looks on different types of devices. You have successfully created an app that can run with a proper UI on phones and tablets that have different screen sizes and densities.