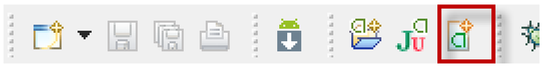
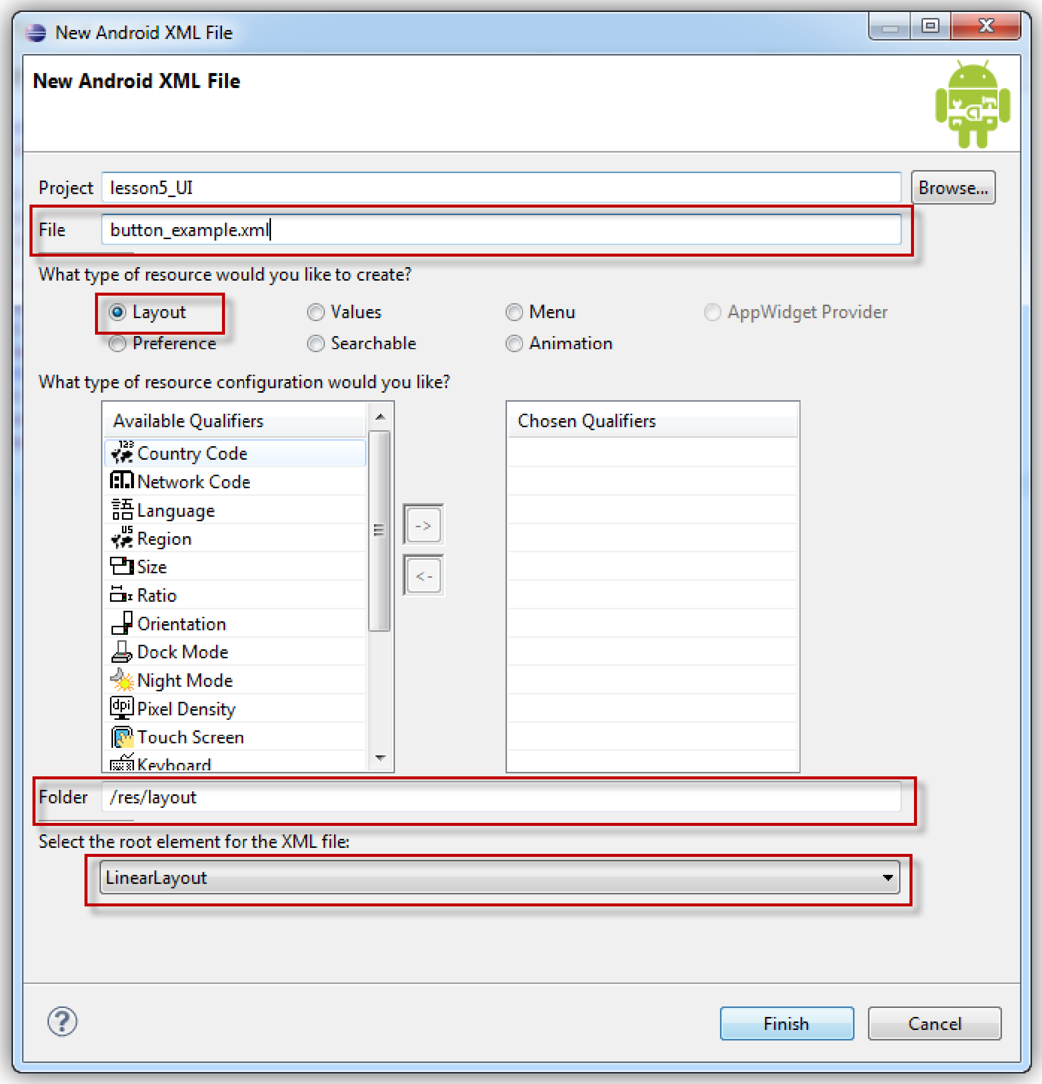
Lesson 5 – User Interfaces Part 2

In this part, we're going to concentrate on Buttons and some of the available events for a button. We're also going to look at a layout property called weight, which allows us to distribute objects by a ratio. In our example, we'll distribute four buttons across a vertical LinearLayout evenly.

Open the Lesson5_UI example, created in Part 1 in Eclipse. Locate the RES folder, and then select the LAYOUT folder. We'll begin by adding a new layout for our button_example.java class that we created in the first part of this lesson.

With LAYOUT selected, select the New File Wizard from the main toolbar:





New Android XML File

New Android XML File

Project: lesson5_UI

File: button_example.xml

What type of resource would you like to create?

(● Layout) ○ Values ○ Menu ○ AppWidget Provider
○ Preference ○ Searchable ○ Animation

What type of resource configuration would you like?

Available Qualifiers: Country Code, Network Code, Language, Region, Size, Ratio, Orientation, Dock Mode, Night Mode, Pixel Density, Touch Screen, Keyboard

Chosen Qualifiers:

Folder: /res/layout

Select the root element for the XML file: LinearLayout

Double check the dialog box to make sure that the following items are selected:

- Project: Lesson5_UI
- File: Type in **button_example.xml**
- Type of resource: Layout is selected
- Folder: /res/layout
- Root Element: LinearLayout

Click Finish. A new XML file will be created for you and opened in Eclipse.

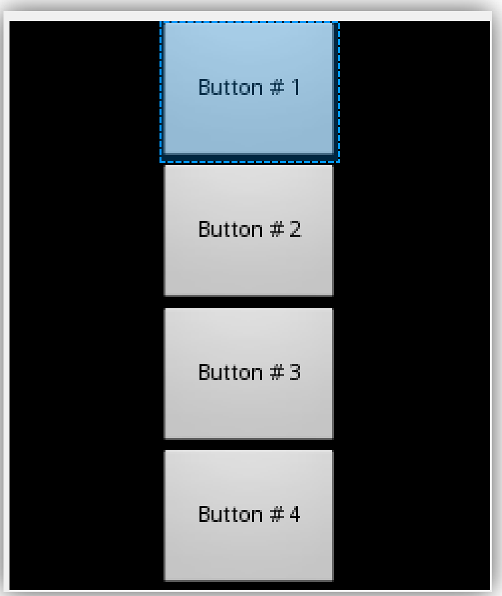
Click on the LinearLayout in the editor screen, and adjust the following properties:
Orientation: Vertical
Gravity: center_horizontal

Drag out 4 buttons to the screen, positioning them one after the other.

For each button, edit the properties as follows:

- Id: @+id/Button01, @+id/Button02, @+id/Button03, @+id/Button04
- Text: Button # 1, Button # 2, Button # 3, Button # 4
- Layout weight: .25 for each button
- Layout width: 120dp

Your screen should now look like this:



Save and close the XML file.

Open the button_example.java class file, and add the following code:

```
public class button_example extends Activity {

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.button_example);

                final Button btn1 = (Button) this.findViewById(R.id.Button01);
                final Button btn2 = (Button) this.findViewById(R.id.Button02);
                final Button btn3 = (Button) this.findViewById(R.id.Button03);
                final Button btn4 = (Button) this.findViewById(R.id.Button04);

                btn1.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {
                                Toast.makeText(getApplicationContext(), "Clicked Button # 1",
                                                Toast.LENGTH_SHORT).show();
                        }
                });
                btn2.setFocusable(false);
                btn2.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {
                                Toast.makeText(getApplicationContext(), "Clicked Button # 2",
                                                Toast.LENGTH_SHORT).show();
                        }
                });

                btn3.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {
                                Toast.makeText(getApplicationContext(), "Clicked Button # 3",
                                                Toast.LENGTH_SHORT).show();
                        }
                });

                btn4.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {
                                Toast.makeText(getApplicationContext(), "Clicked Button # 4",
                                                Toast.LENGTH_SHORT).show();

                        }
                });
                btn4.setOnFocusChangeListener(new View.OnFocusChangeListener() {

                        @Override
                        public void onFocusChange(View v, boolean hasFocus) {
                                if (hasFocus) {
                                        Toast.makeText(getApplicationContext(), "Has Focus",
                                                        Toast.LENGTH_SHORT).show();
                                } else {
                                        Toast.makeText(getApplicationContext(), "Lost Focus",
                                                        Toast.LENGTH_SHORT).show();
                                }
                        }
                });
                btn4.setOnLongClickListener(new View.OnLongClickListener() {

                        @Override
                        public boolean onLongClick(View v) {
                                Toast.makeText(getApplicationContext(),
                                                "Long Press Button # 4", Toast.LENGTH_SHORT).show();
                                return false;
                        }
                });

        }

}
```

We’ve added the following functionality:

1 – Each button has a click event, which will display a short “toast” message on the screen when clicked.

2 – Button # 2 is not focusable. You can see this in the emulator by enabling the scroll wheel. Press F6 to enable scrolling and move the mouse around. You’ll see that Button # 1, 3, and 4 can turn orange (have focus), but not #2. That was achieved with the setFocusable(false) code for btn2.

3 – Button # 4 has several characteristics available to it. First, an event is triggered for the button’s focus state. You’ll see “Has focus” and “Lost focus” messages when the button receives and loses focus. Long pressing on Button # 4 generates another event as well.

Review the video that demonstrates this functionality.