Tutorial 4 Setup Main & Query Pages

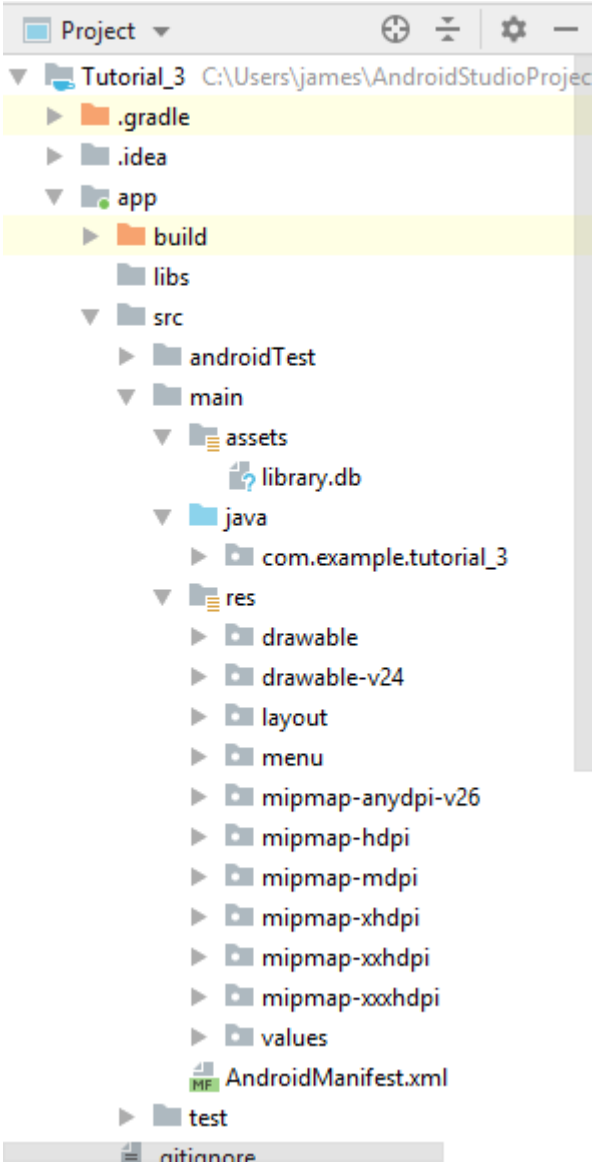
In Tutorial 3, we started with creating a library project in Android development environment. We will be continuing to improve our project by adding main and query pages in this tutorial. Note--I created a new project called Tutorial 4. The xml files from Tutorial 4 can be copied to a later tutorial project, just like you can copy the java classes from Tutorial 3.

Contents

Background for the Exercise:

• Android's File/Folder Hierarchy and Structure

You should place each type of resource or files in a specific directory of your project. For example, here's the file hierarchy for a simple project:



As you can see in this example,

- res/ is the resource directory. Resources in Android are files stored under the res directory of your project. Resources can be physical files (Audio, video, images, text, etc…) or xml files that declare some values to use in your application.
  - Strings, colors, arrays, dimensions. Defined in res/values/ directory. Using them is very useful in Localization and Internationalization.
  - Images put in res/drawable directory. You can put all the images or icons you need in your application.
  - Layout resources, defined in res/layout/ for declaring the views that construct the user interface of the activities.[1]
- assets/ is the resource directory. We will only store our database in this directory. The difference between assets/ and res/ directory is that assets/ directory resource file will not be automatically generated. So if you need to read assets directory file in your application, you must specify the path to the file.
- src/ is the original coding storage directory.

[1] Samy, Mina. (2011). Android App Development – Using Android resources part 1: String Resources. from http://mobileorchard.com/android-app-development-using-android-resources-part-1-string-resources/

- AndroidManifest.xml is the project's list file
  This file lists the functions in the application. When you need to use a new function or activity file, you must register the new function or activity in this file. Besides, in this file, you can add authority to your application.

What is a Layout?

An Android layout is a class that handles arranging the way its children appear on the screen. Anything that is a View (or inherits from View) can be a child of a layout. All of the layouts inherit from ViewGroup (which inherits from View) so you can nest layouts. You could also create your own custom layout by making a class that inherits from ViewGroup.

The standard Layouts are: AbsoluteLayout, FrameLayout, LinearLayout, RelativeLayout and TableLayout.[2]

In this tutorial we will use LinearLayout. It displays child View controls in a single row or column. It is a very handy layout method for creating forms.

Android Basic Widgets

Buttons

Android SDK includes two simple button controls for use within your layouts: Button (android.widget.Button) and ImageButton (android.widget.ImageButton). The Button control has a text label, whereas the ImageButton uses an image drawable resource instead.

TextView

The Android SDK includes a simple static text control for use within your layouts: TextView (android.widget.TextView). A good example of TextView control usage would be to display textual labels for other controls, like "Enter a Date:", "Enter a Name:" or "Enter a Password:".[3]

[2] Android, Learn. (2010). Tutorials for Developing with Android. from http://www.learn-android.com/2010/01/05/android-layout-tutorial/

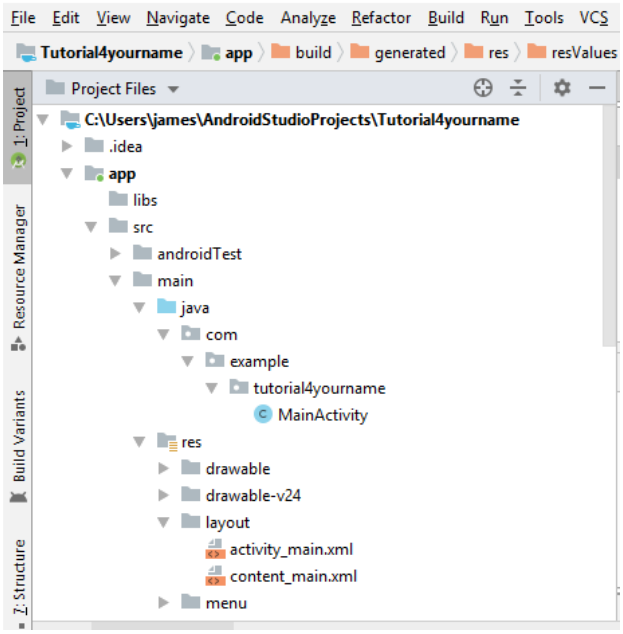
[3] Tuts, Mobile. (2010). Android User Interface Design: Basic Buttons. from http://mobile.tutsplus.com/tutorials/android/android-interface-design_basic-buttons/

Exercise

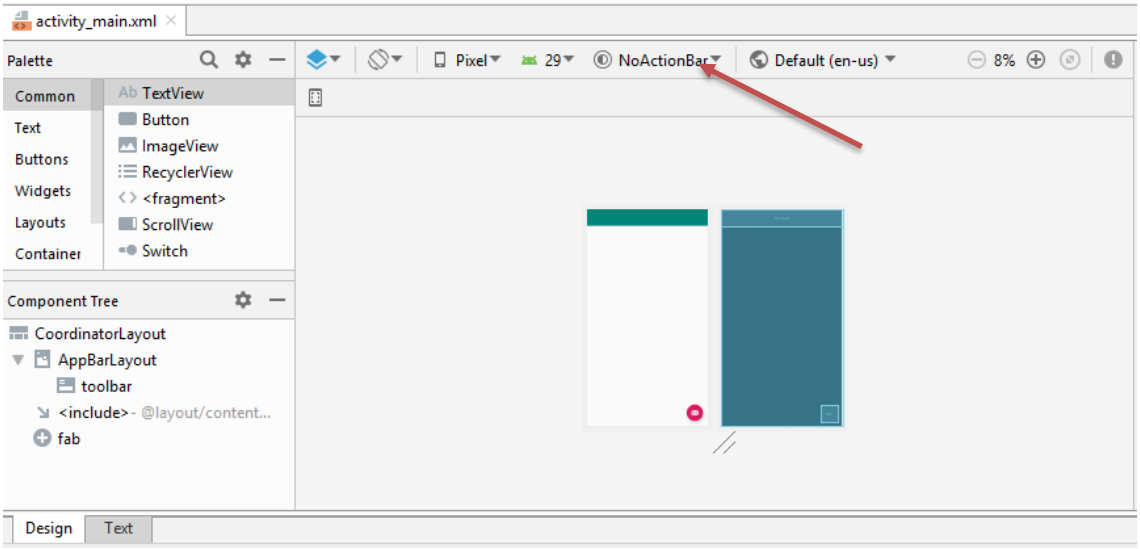
I. Set up Main page

1. Find the .xml file

Android User Interfaces (UI) are defined in xml files. When you create a new project, **main.xml** will be created automatically. Expand **Tutorial4_YourName** project-> **res** -> **layout**. In this project, we created the main.xml as the name of activity_main.xml.

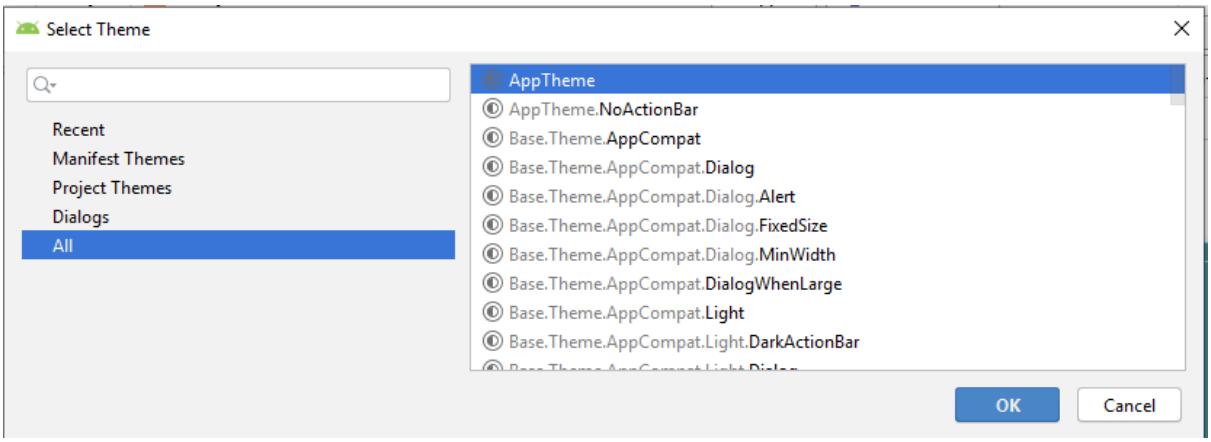


Double click **activity_ main.xml**. Now we can start the tour.



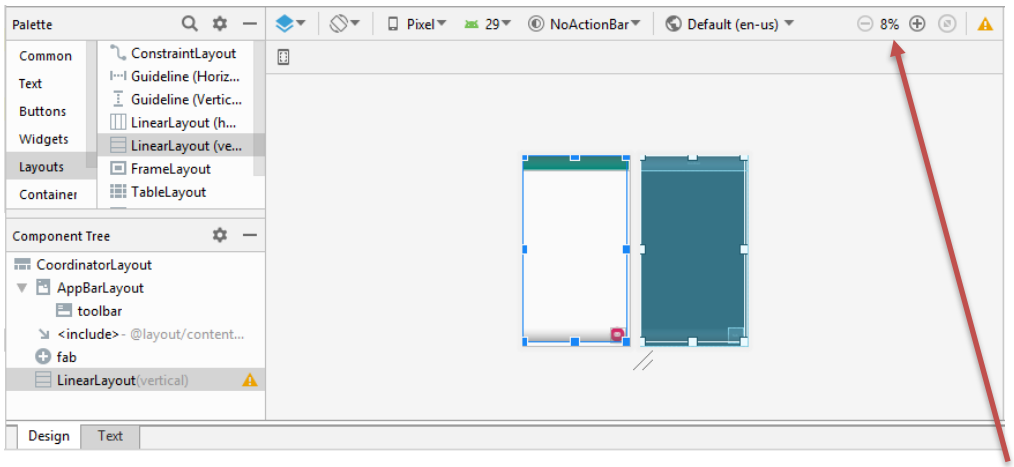
2. Choose the theme.

We`ll use default NoActionBar theme here, but you can also explore different themes in **More Themes** list as shown below. Prior screen shows the drop-down box to change themes.



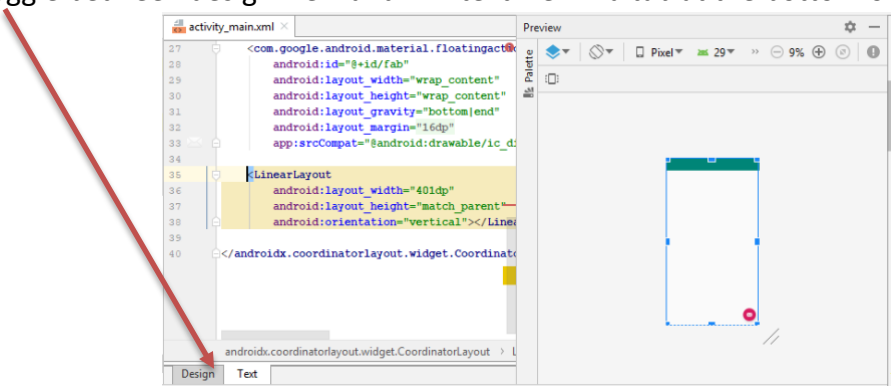
3. Set up layouts

✓ Now, expand **Layouts** under Palette and drag **LinearLayout (Vertical)** to layout editor's front panel.



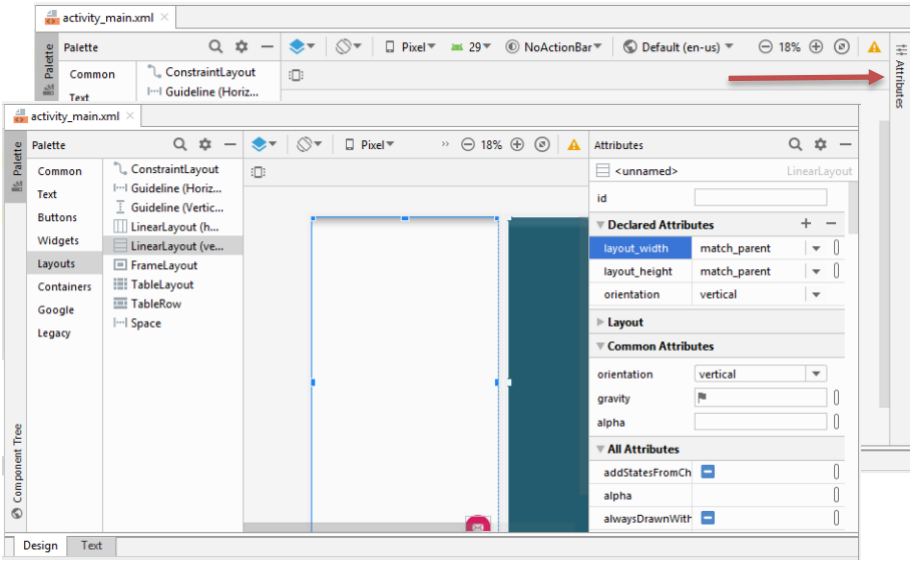
✓ Since we`ll use **LinearLayout** as a fundamental layout, let`s make it bigger. Use the – and + to on the command bar (i.e., far right side) to increase/decrease the design view.

✓ Toggle between design view and xml text view via tab at the bottom of the panel.



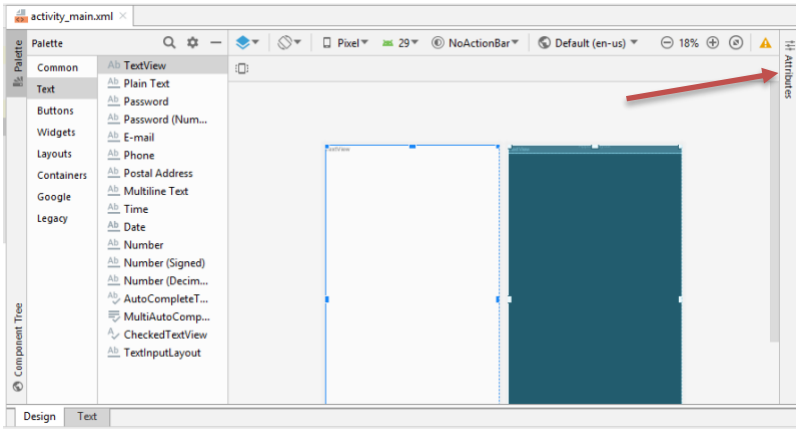
✓ Edit layout_width and layout_height directly in the XML file

✓ Click to expand the Attributes panel on far right to enter layout_width / layout_height directly.



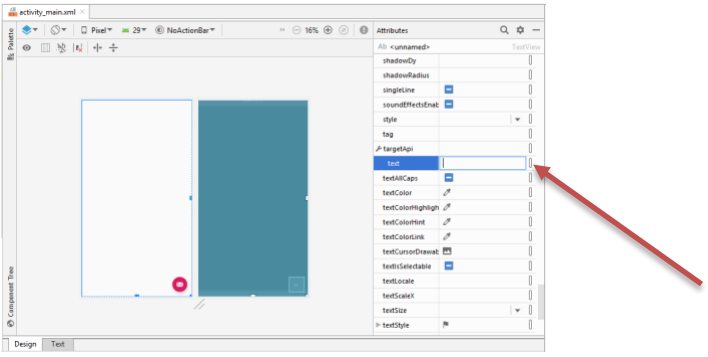
4. Add Title

✓ Expand **Palette Text** -> choose **TextView** -> drag it into layout editor.

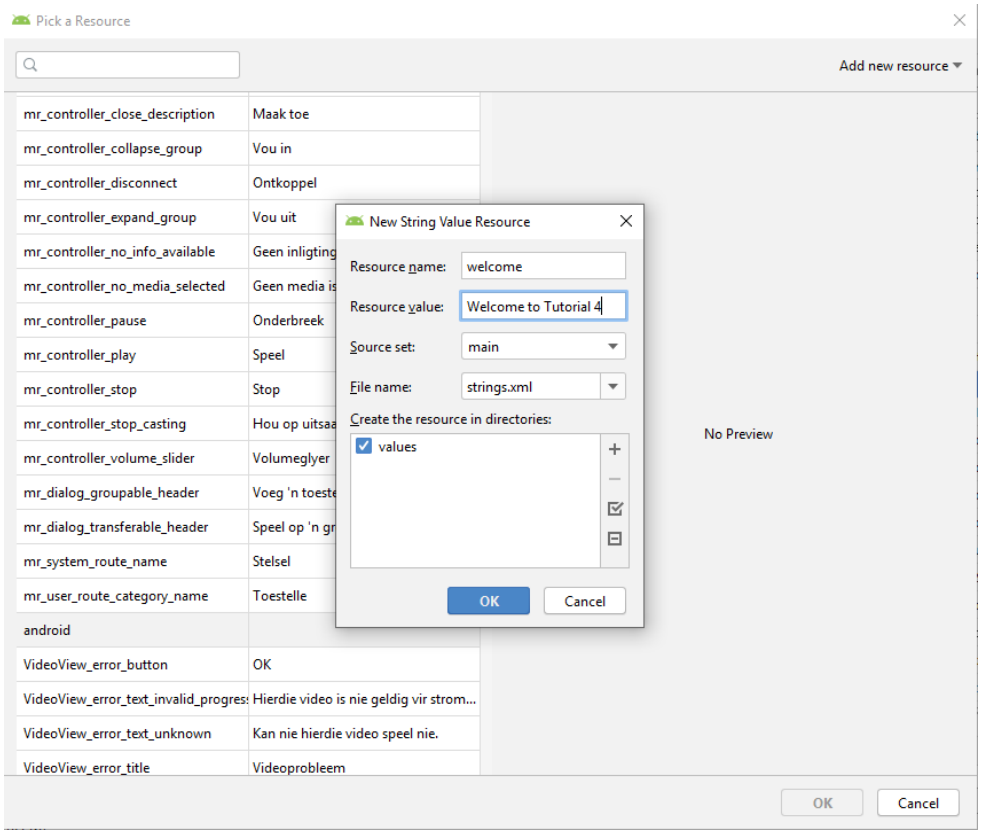


✓ Now let`s change the text.

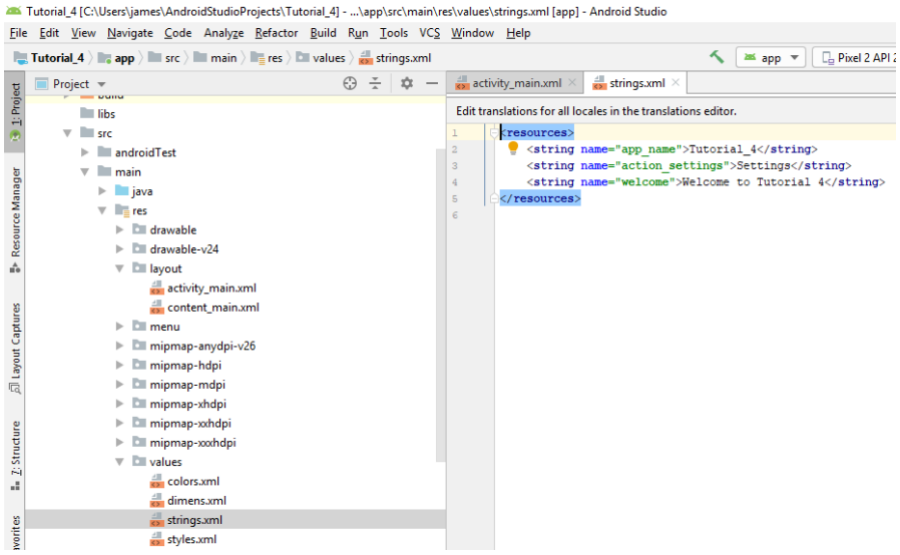
- Expand the attributes panel and scroll down to text, then click on the pick a new resource icon…



- Choose Add new resource…then fill in the pop-up window as below…



- You have created a new string named welcome and you can see its value in the strings.xml file under values.



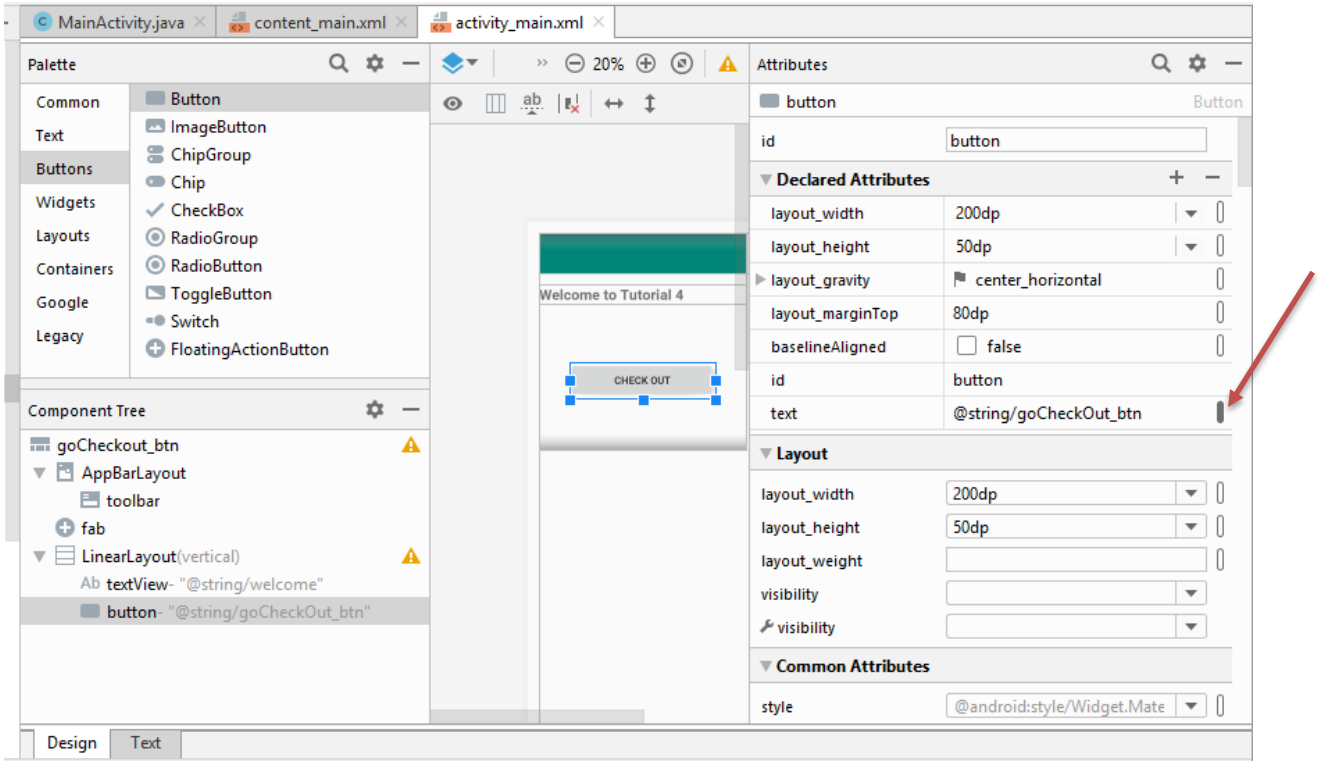
✓ Set up the properties of the **TextView**. Use these values as properties for reference. You can change the value according to your design. Please change these settings in the Attributes panel.

- Text size: 20sp
- Text style: bold
- Layout gravity: center horizontal

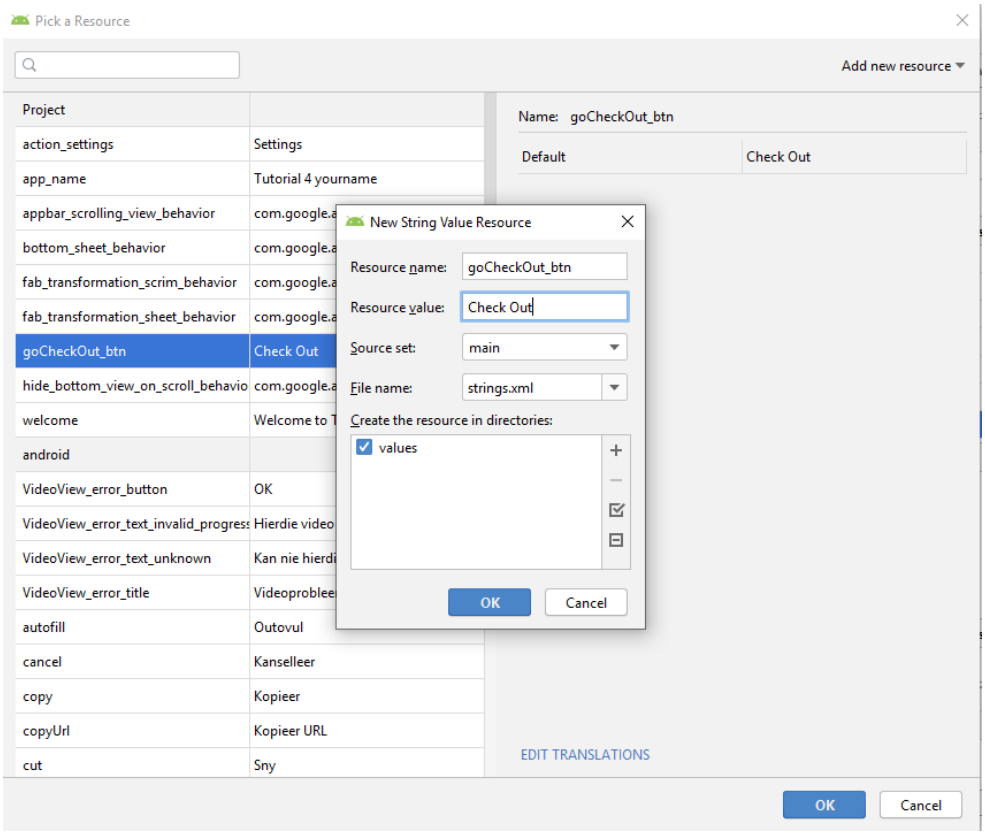
if you follow all steps, the UI should look like this:

5. Add Button

✓ In Design Palette, click on **Buttons** and drag **Button** to layout editor and put it below the title we just added.



✓ Use the Pick a Resource icon to add a new string variable goCheckOut_btn and the value being Check Out. The picture below shows the Pick a Resource screen with the goCheckOut_btn variable name and variable value being Check Out. The screen above shows the button with the properties on the right and the button under the title on the left side of the properties.



✓ Set the value of the button properties via Attributes to the following as noted on the prior attributes picture:

- Text style: bold
- Layout gravity: center horizontal
- Layout margin top: 80dp
- Width: 200dp

✓ Now the UI is supposed to look like this:

6. Do it yourself

Add a query button, just like the previous button.
Recommended attribute properties are:

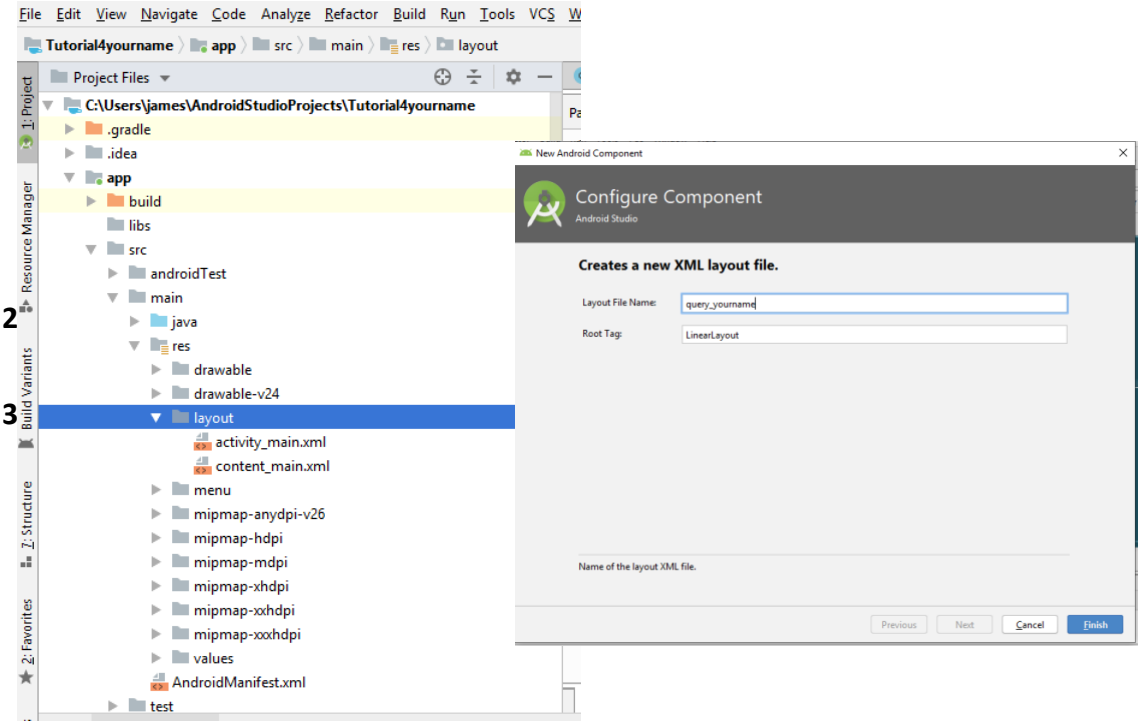
- ID: goDoQuery_btn
- Text style: bold
- Layout gravity: center horizontal
- Layout margin top: 100dp
- Layout height: wrap_content
- Layout width: 200dp
- String: Do Query
- R.String: goDoQuery_btn

II. Set Up Query page

1. Create a new .xml file

Right click **layout** in Package Explorer -> **New** -> **XML** -> **Layout XML File**

Name the new .xml file as query_yourname and click **finish**.



2. Set up layouts

Same layout format as **activity_ main.xml**

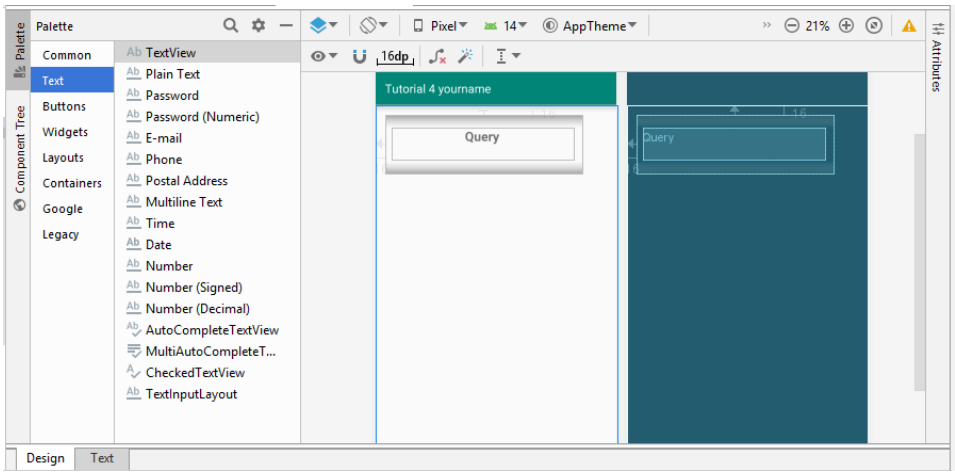
3. Add title

- String: Query

R.string: title_query

- Text size: 22sp
- Text style: bold
- Layout gravity: center horizontal
- Layout margin top: 15dp

The Design Palette Panel should look like this:



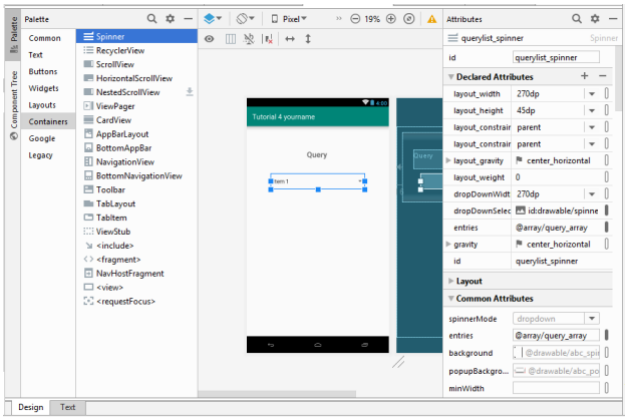
4. Add drop-down list button

✓ In Design Palette, choose **Containers** and drag **Spinner** to layout editor and put it below the title.

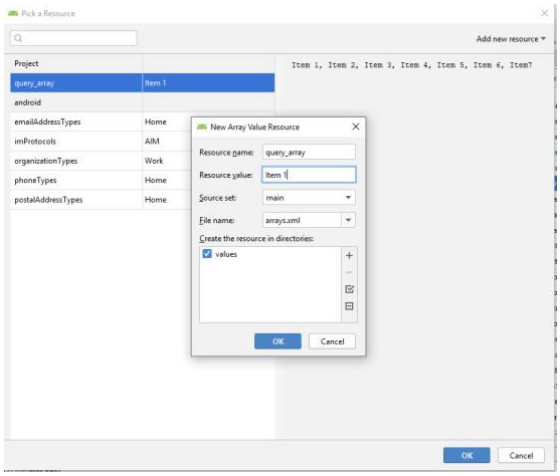
✓ Set up layout properties

- Layout gravity: center horizontal
- Layout height: 45dp
- Layout margin top: 15dp
- Layout width: 270dp
- ID querylist_spinner
  - XML:@id/querylist_spinner
  - Java: R.id.querylist_spinner

✓ Now the Design Palette Panel should look like:



✓ We also want to change the names of entries into the spinner. In Attributes, find entries for querylist_spinner and use the Pick a Resource icon to create an array. Likewise, locate prompt in Attributes and use Pick a Resource to add a query_prompt string with value of Choose a Query.

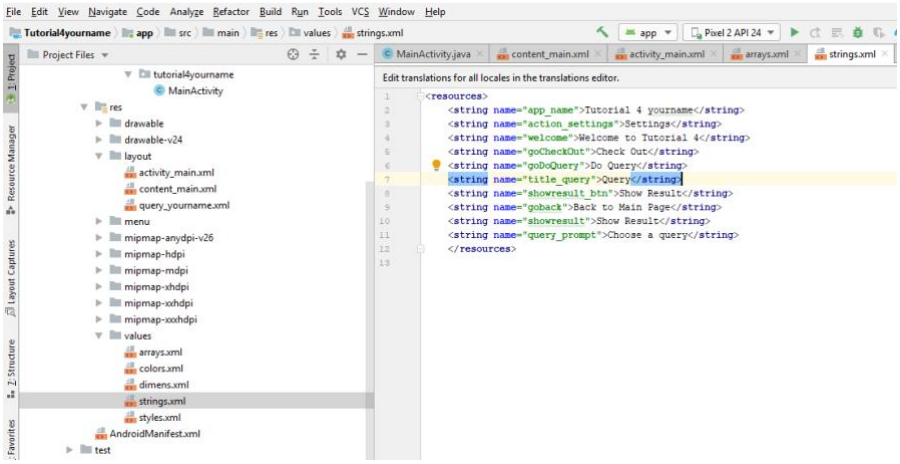


✓ To complete the array items, open **arrays.xml** in Package Explorer **-> src -> main -> res** -> **values** and change the xml file code to read as below, to create seven items in the query_array. Pick a Resource created arrays.xml and query_array.

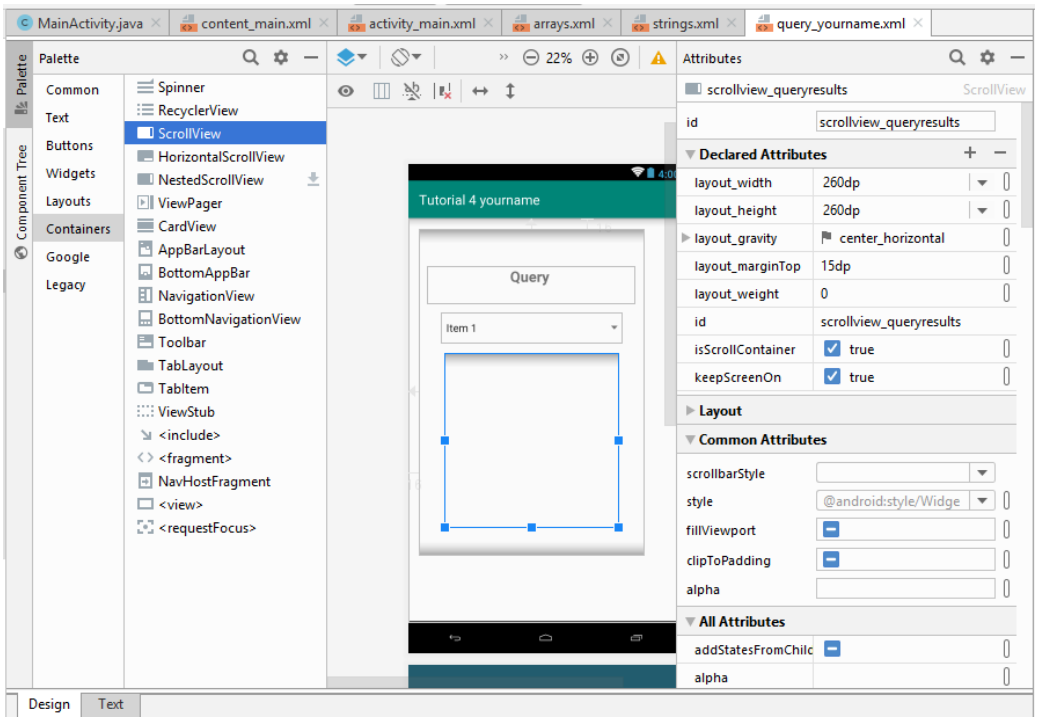
MainActivity.java | content_main.xml | activity_main.xml | arrays.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3  <string-array name="query_array">
4      <item>Query 1</item>
5      <item>Query 2</item>
6      <item>Query 3</item>
7      <item>Query 4</item>
8      <item>Query 5</item>
9      <item>Query 6</item>
10     <item>Query 7</item>
11 </string-array>
12
13 </resources>
```

✓ Open **strings.xml** in Package Explorer **-> src -> main -> res** -> **values** to identify all the string variables. As you add string variables via Pick a Resource, the variables and their values are added to strings.xml



5. **Add ScrollView to show results of queries...**
   - ✓ In Design Pallete Panel click on **Containers** and drag **ScrollVIew** to layout editor
   - ✓ Set up layout properties:
     - Layout gravity: center horizontal;
     - Layout height: 260dp;
     - Layout width: 260dp
     - ID = scollview_queryresults
   - ✓ The UI should look like this:



6. **Do it yourself**

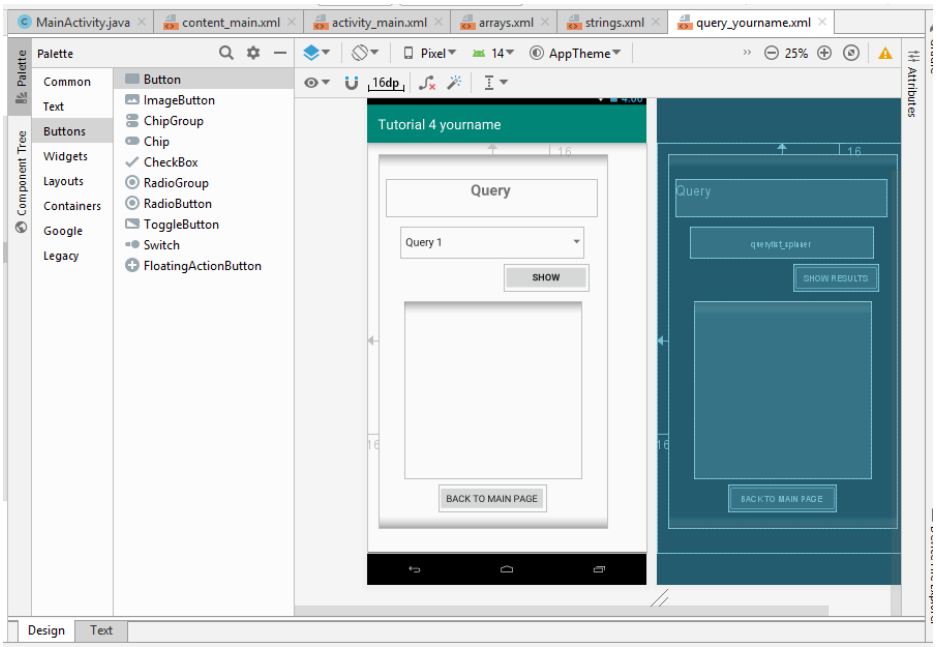
   Set up other widgets and finish the page

   1) "Show Result" Button
   - ID: showresult_btn

- String: Show Result
  R.string: showresult
- Text size: 14sp
- Text style: Bold
- Layout gravity: right|center horizontal
- Layout width: 125dp
- Layout height: 40dp
- Layout margin right: 26dp
- Layout margin top: 10dp

2) "Back to Main Page" Button

- ID: goback_btn
- String: Back to Main Page
  R.string: goback
- Text size: 14sp
- Text style: Bold
- Layout gravity: center horizontal
- Layout width: wrap_content
- Layout height: 40dp
- Your Design Panel should look like:



What to Submit

Two Screenshots

Please capture the screenshots of your completed Query page and paste them into a Word document.

The screenshots should look like:

All deliverables should be submitted via the Canvas assignment file upload as a single Word or PDF document by the due date.

Additional Examples (Optional)

Note** Android Studio 3.5 may have slightly different screens

If you are interested in learning more about Android Application User Interface design, here is more information you might find helpful.

In the tutorial, we used a drag & drop modeling editor to visually design user interfaces.

There are other two ways to create UI components. The most convenient and maintainable way to design application user interfaces is creating XML layout resources. You can also programmatically create UI components in JAVA. In this optional reading material, we will introduce how to design UI by creating XML layout resources.

What you can learn from this example:

- What is the difference between the drawable-ldpi, drawable-mdpi, and drawable-hdpi in the res directory?
- Customizing the Styling, Size and Font of a TextView Control
- Controlling the Background of a TextView Control

Steps:

1. Create a new project, name it : UIExample01

2. Save a picture you like in res/drawable-hdp for the background use. For Example：

Tips:

Drawable(hdpi,ldpi,mdpi) folder is used to save pictures.
Hdpi is for high resolution ratio picture, like WVGA (480×800) and FWVGA (480×854).
mdpi is for middle resolution ratio picture, like HVGA (320×480).
ldpi is for low resolution ratio picture, like QVGA (240×320).
In common， there won't be any problem to save pictures in hdpi.

.WVGA,HVGA,QVGA:
VGA (Video Graphics Array) - 640*480 pixels.
WVGA(Wide VGA) - 480*800 pixels.
HVGA(Half VGA) - 320*480 pixels.
QVGA(Quarter VGA) - 240*320 pixels.

3. Add the following codes into res/layout/activity_main.xml：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@drawable/background" >

    <TextView
        android:id="@+id/blackTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/blueTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/dkgrayTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

```
    <TextView
        android:id="@+id/grayTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/greenTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/ltgrayTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/magentaTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/redTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/transparentTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/whiteTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/yellowTextView"
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"/>

</LinearLayout>
```

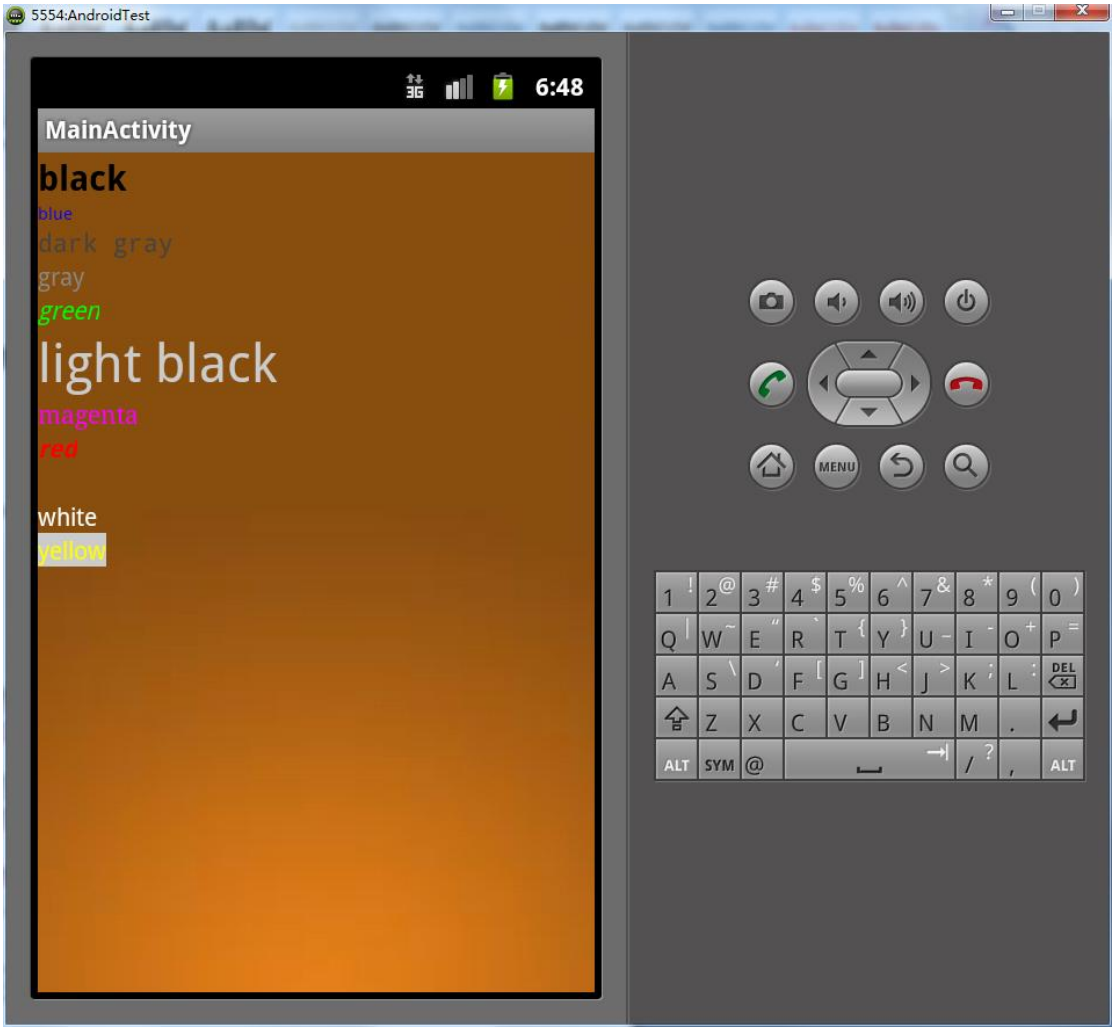
4. Add the following codes into src/com.example.uiexample01/MainActivity.java:

```
package com.example.uiexample01;

import android.os.Bundle;
import android.app.Activity;
import android.graphics.Color;
import android.graphics.Typeface;
import android.view.Menu;
import android.widget.TextView;

public class MainActivity extends Activity {
    private TextView blackTextView,blueTextView,dkgrayTextView,grayTextView,
            greenTextView,ltgrayTextView,magentaTextView,redTextView,
            transparentTextView,whiteTextView,yellowTextView;

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

        getTextView();
        setTextViewText();
        setTextViewColor();
        setTextViewTextSize();
        setTextStyle();
    }

    public void getTextView(){
        //Get the instances of each TextView from the layout
        blackTextView=(TextView) this.findViewById(R.id.blackTextView);
        blueTextView=(TextView) this.findViewById(R.id.blueTextView);
        dkgrayTextView=(TextView) this.findViewById(R.id.dkgrayTextView);
        grayTextView=(TextView) this.findViewById(R.id.grayTextView);
```

```
            greenTextView=(TextView) this.findViewById(R.id.greenTextView);
            ltgrayTextView=(TextView) this.findViewById(R.id.ltgrayTextView);
            magentaTextView=(TextView) this.findViewById(R.id.magentaTextView);
            redTextView=(TextView) this.findViewById(R.id.redTextView);
            transparentTextView=(TextView)
this.findViewById(R.id.transparentTextView);
            whiteTextView=(TextView) this.findViewById(R.id.whiteTextView);
            yellowTextView=(TextView) this.findViewById(R.id.yellowTextView);
      }

      public void setTextViewText(){
            //Set the text on the TextView
            blackTextView.setText("black");
            blueTextView.setText("blue");
            dkgrayTextView.setText("dark gray");
            grayTextView.setText("gray");
            greenTextView.setText("green");
            ltgrayTextView.setText("light black");
            magentaTextView.setText("magenta");
            redTextView.setText("red");
            transparentTextView.setText("transparent");
            whiteTextView.setText("white");
            yellowTextView.setText("yellow");
      }

      public void setTextViewTextSize(){
            //Set the size of text on the TextView
            blackTextView.setTextSize(20);
            blueTextView.setTextSize(10);
            ltgrayTextView.setTextSize(30);
      }

      public void setTextViewColor(){
            //Set the color of text on the TextView
            blackTextView.setTextColor(Color.BLACK);
            blueTextView.setTextColor(Color.BLUE);
            dkgrayTextView.setTextColor(Color.DKGRAY);
            grayTextView.setTextColor(Color.GRAY);
```

```
            greenTextView.setTextColor(Color.GREEN);
            ltgrayTextView.setTextColor(Color.LTGRAY);
            magentaTextView.setTextColor(Color.MAGENTA);
            redTextView.setTextColor(Color.RED);
            transparentTextView.setTextColor(Color.TRANSPARENT);
            whiteTextView.setTextColor(Color.WHITE);
            yellowTextView.setTextColor(Color.YELLOW);

            //Set the color of background of TextView itself
            yellowTextView.setBackgroundColor(Color.LTGRAY);
        }

        public void setTextStyle(){
            //Set the text style
            blackTextView.setTypeface(Typeface.DEFAULT_BOLD);
            blueTextView.setTypeface(Typeface.DEFAULT);
            dkgrayTextView.setTypeface(Typeface.MONOSPACE);
            ltgrayTextView.setTypeface(Typeface.SANS_SERIF);
            magentaTextView.setTypeface(Typeface.SERIF);
            greenTextView.setTypeface(null, Typeface.ITALIC);
            redTextView.setTypeface(null, Typeface.BOLD_ITALIC);
        }
}
```

5. The result should look like：



Explanation:

Adding a TextView Control to a Layout:

```
android:id="@+id/blackTextView"
```

When you later want to use this widget, you can find it by its id, like:

```
TextView blackTextView=(TextView) this.findViewById(R.id.blackTextView);
```

Controlling the Background of a TextView Control:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
    android:background="@drawable/background" >
……
```

Set the properties of the textview:

```
blackTextView.setText("black");
```

Customizing the Font of a TextView Control:

```
blackTextView.setTextSize(20);
```

Customizing the Text Color of a TextView Control:

```
blackTextView.setTextColor(Color.BLACK);
```

Customizing the Styling of a TextView Control:

```
blackTextView.setTypeface(Typeface.DEFAULT_BOLD);
```