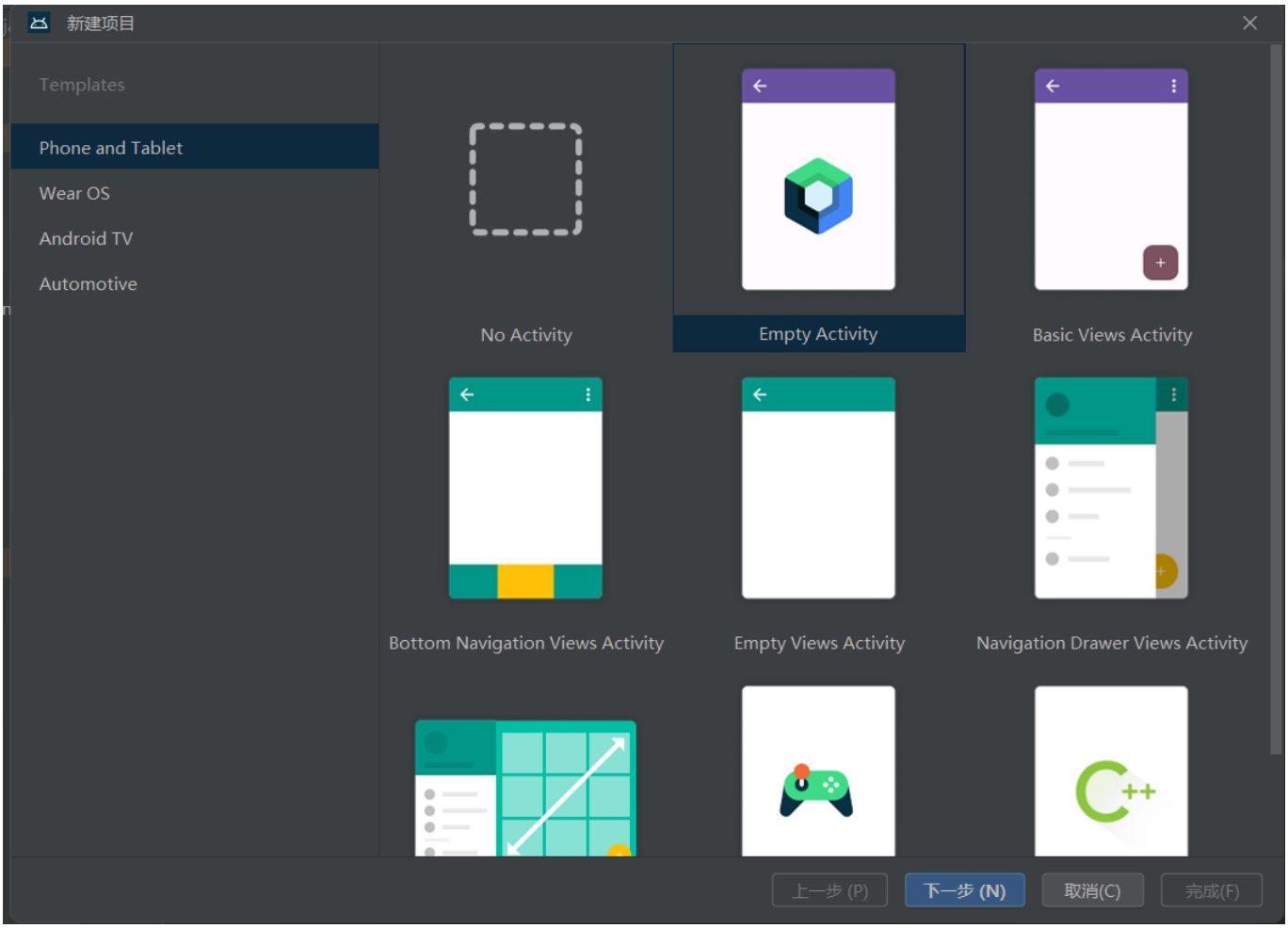
实验二 构建第一个Kotlin应用

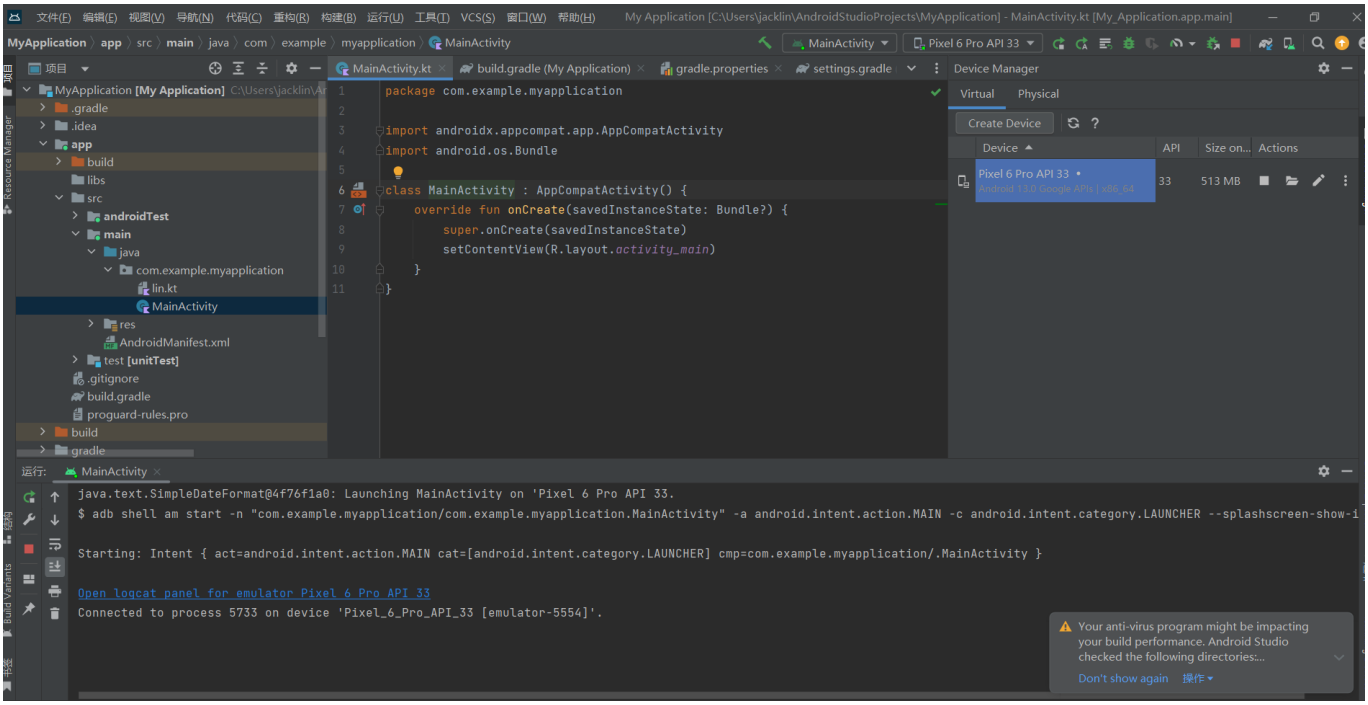
创建第一个Kotlin应用

打开Android Studio，选择Projects>New Project，然后选择Basic Activity.



1、Android Studio的界面布局

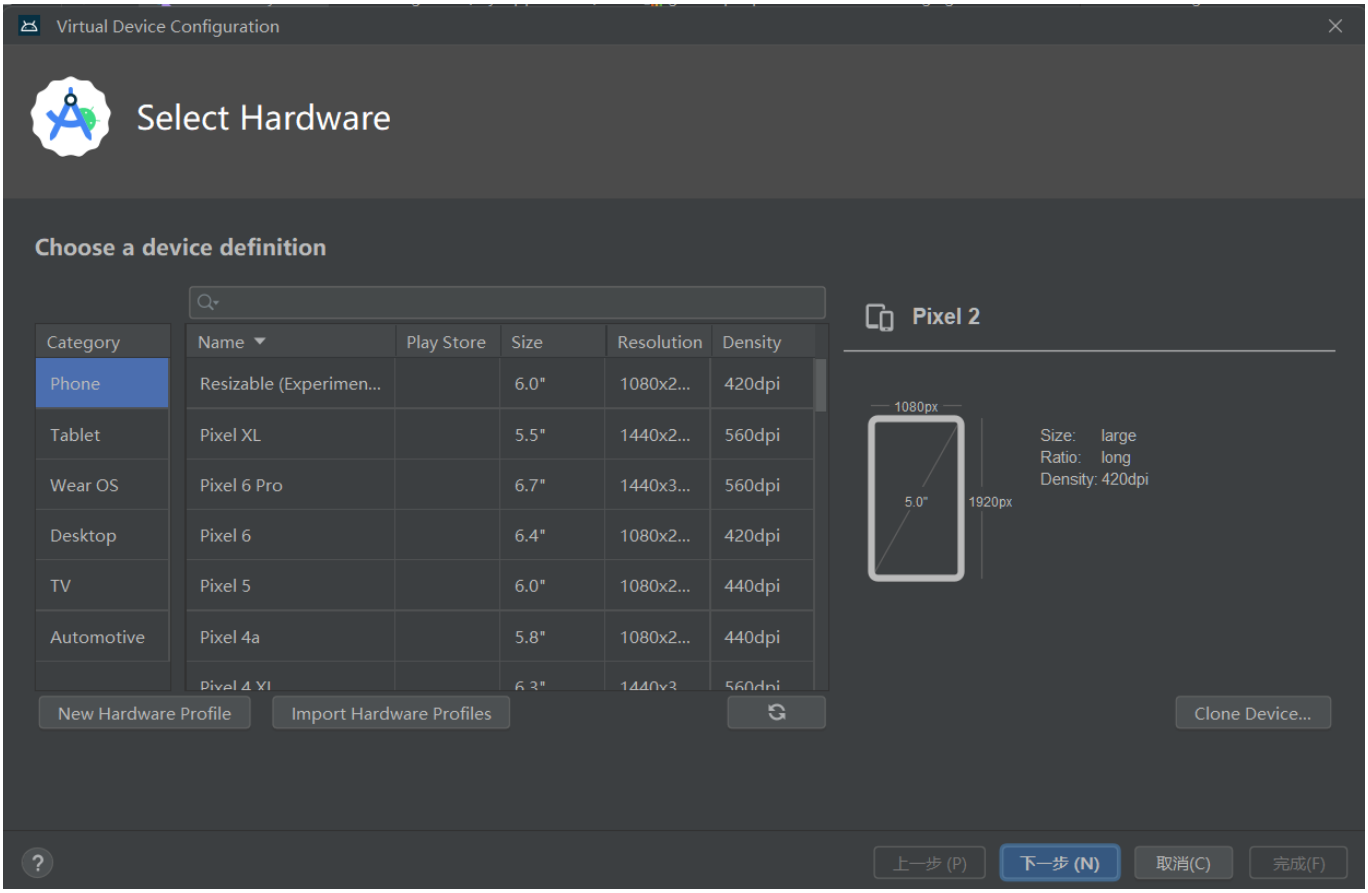
Android Studio工作区:



2、创建模拟器

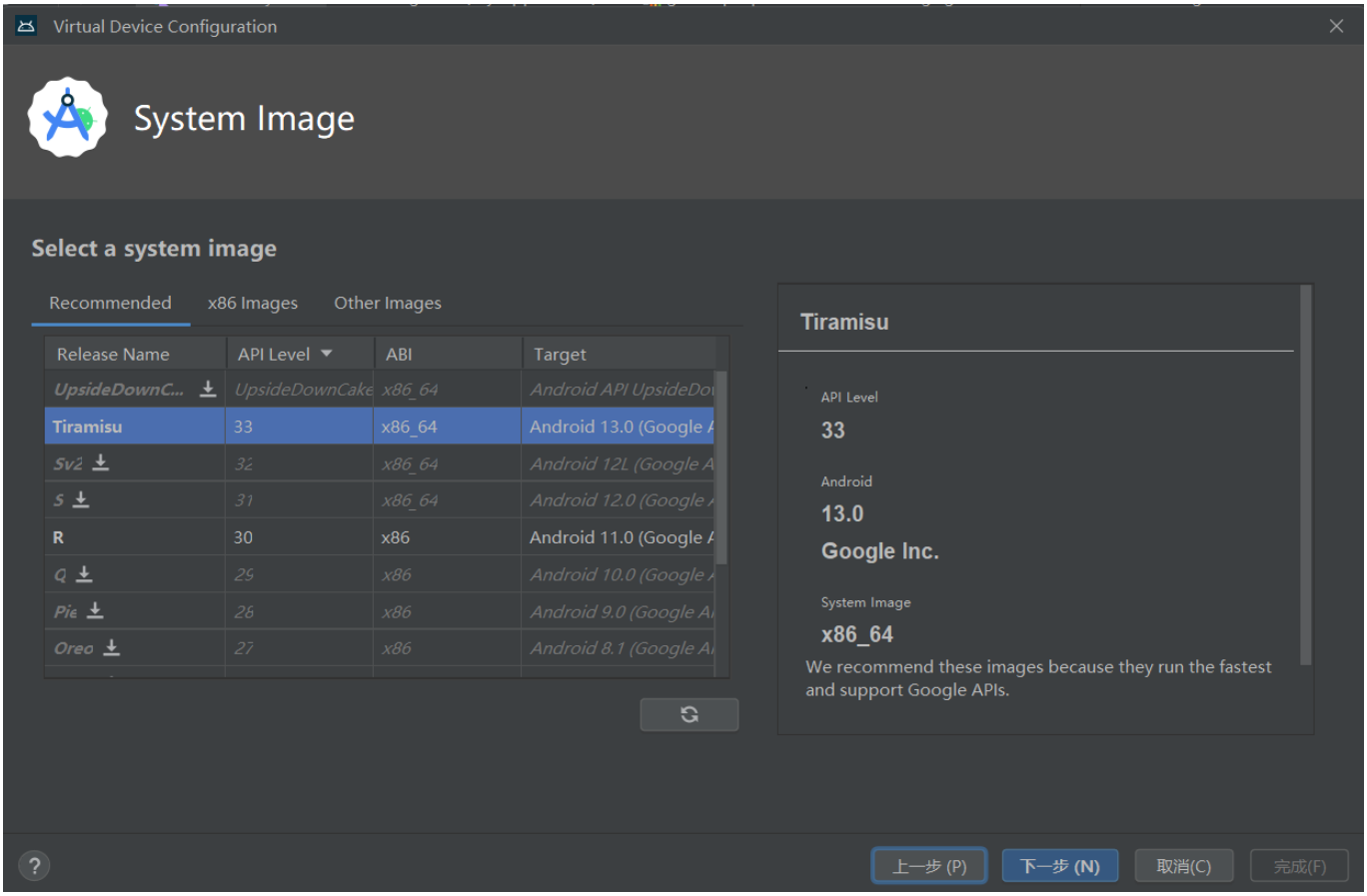
创建可以运行APP的模拟器

点击Create device，弹出创建模拟器的页面



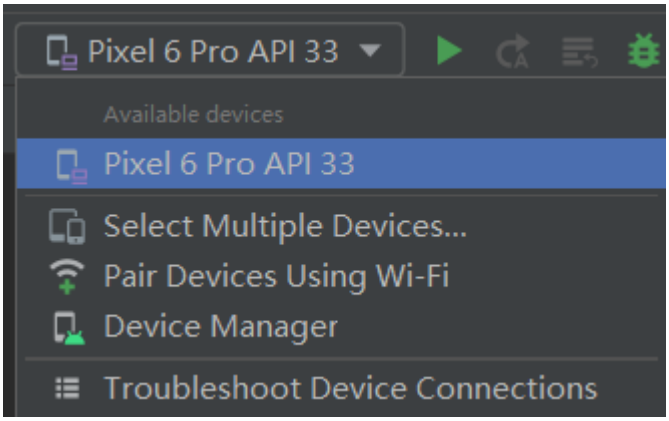
下载镜像

在系统镜像页面的Recommended标签栏，选择最新镜像

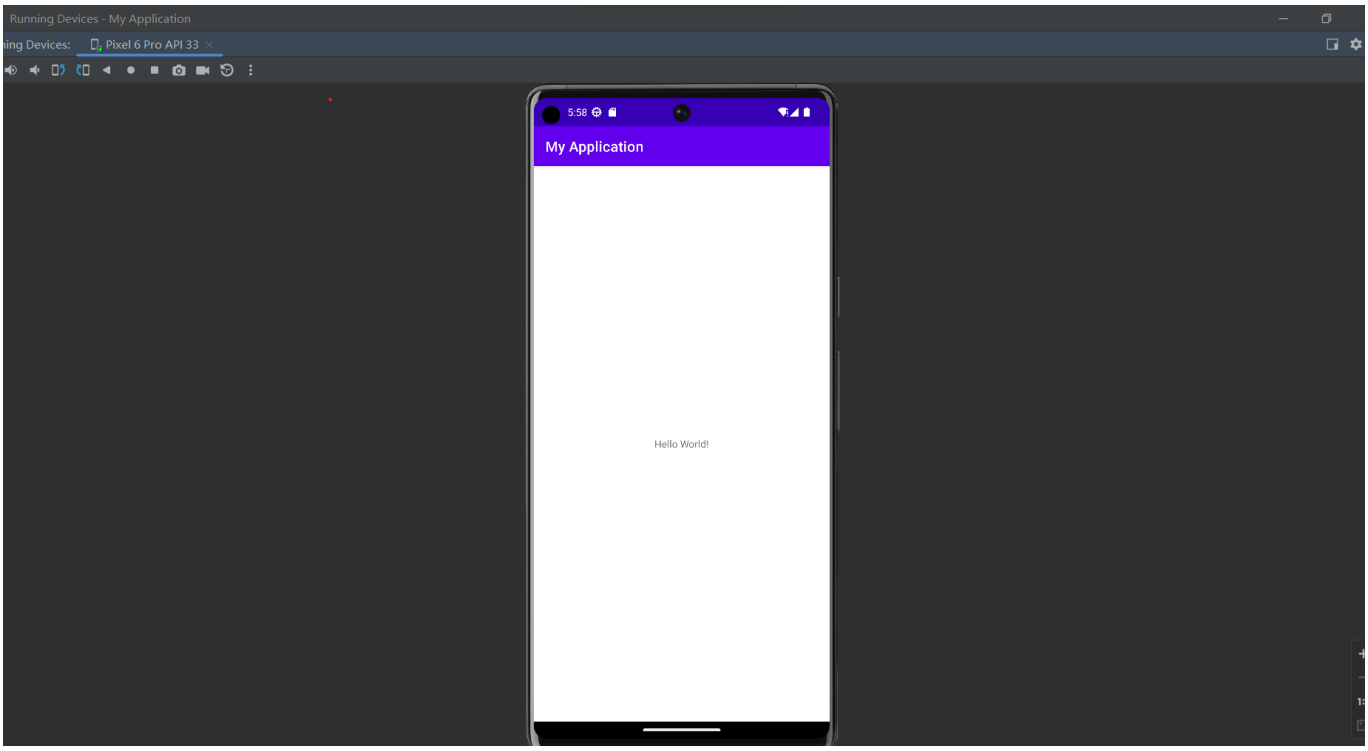


在模拟器上运行应用程序

选择Run>Run ‘app’，在工具栏上可以看到运行程序的一些选择项



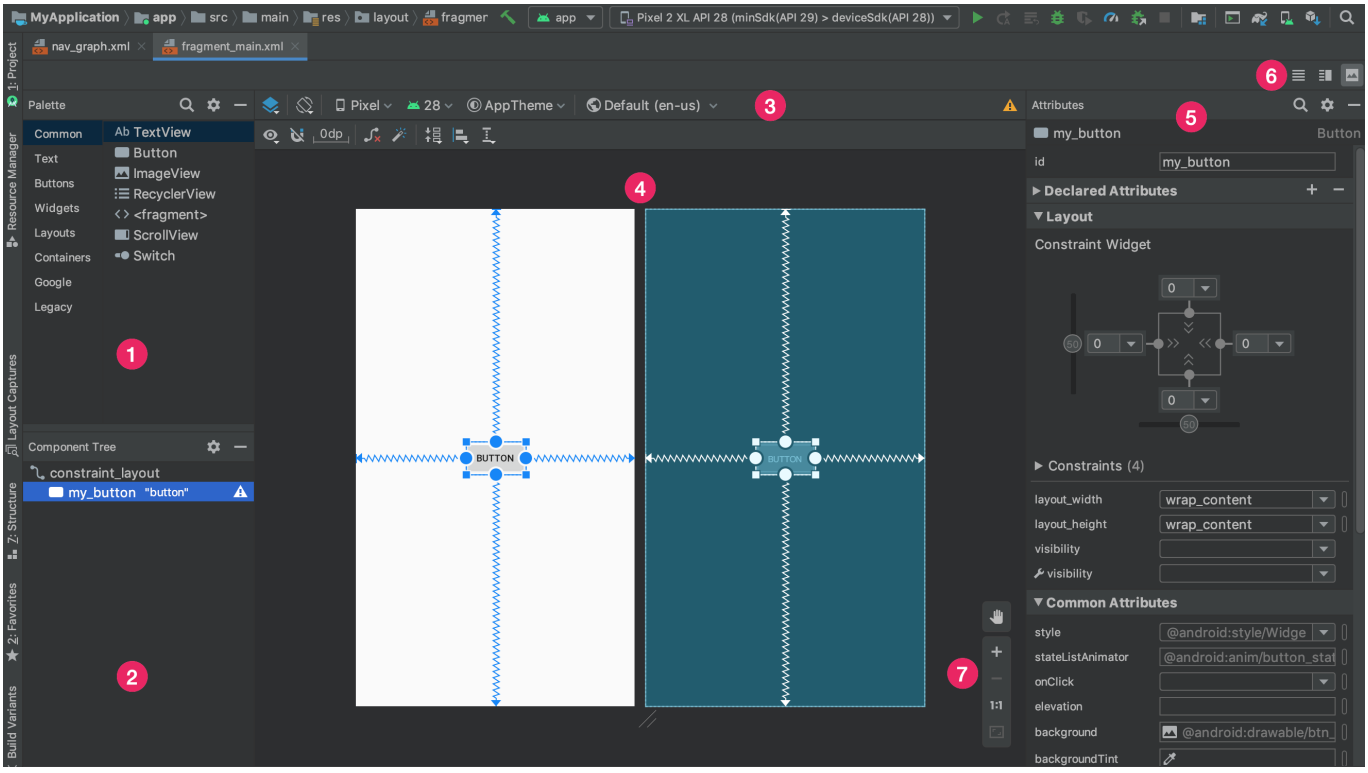
运行效果：



3、查看布局编辑器

什么是布局编辑器

当我们打开 XML 布局文件时，就会显示布局编辑器，如下图：

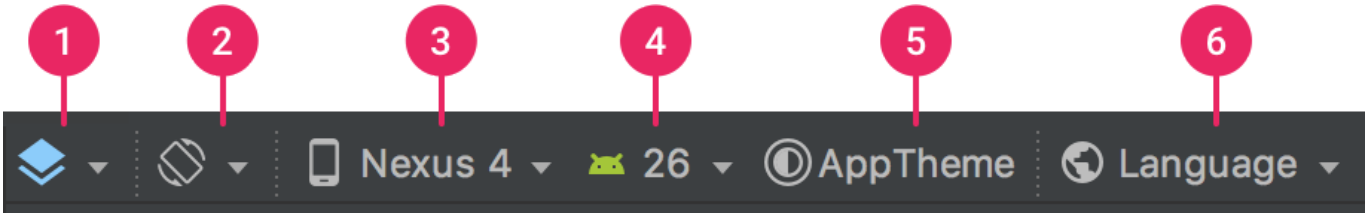


1. Palette：包含可以拖到布局中的各种视图和视图组
2. Component Tree：显示布局中的组件层次结构
3. 工具栏：点击这些按钮可在编辑器中配置布局外观及更改布局属性
4. 设计编辑器：在 Design 视图和/或 Blueprint 视图中修改布局
5. Attributes：用于对所选视图的属性进行控制的控件

6. 视图模式：采用 Code 模式、Design 模式或 Split 模式模式查看布局。Split 模式会同时显示 Code 和 Design 窗口
7. 缩放和平移控件：控制编辑器内的预览大小和位置

如何更改预览外观

Android 布局编辑器支持在不同配置下的预览，我们可以使用设计编辑器顶行中的按钮在编辑器中配置布局的外观。



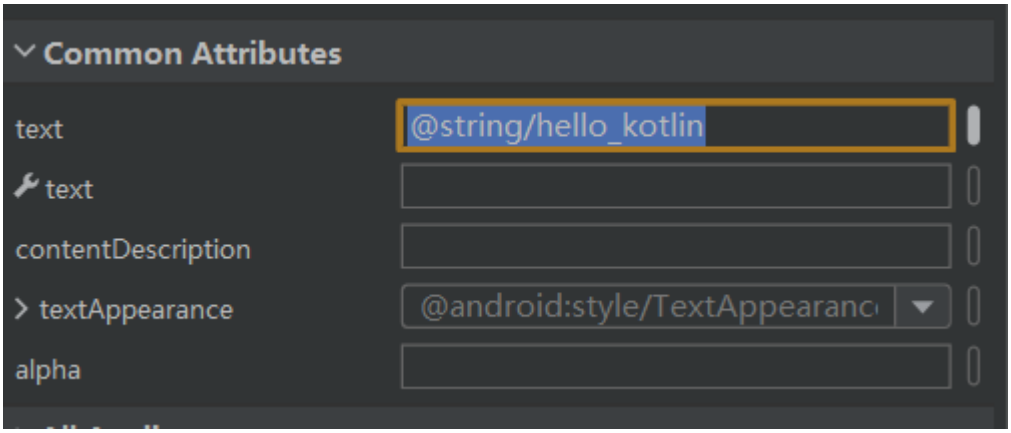
1. Design 和 Blueprint：选择我们希望如何在编辑器中查看布局。选择 Design 可查看布局的渲染后预览效果。选择 Blueprint 可仅查看每个视图的轮廓。选择 Design + Blueprint 可并排查看这两个视图；
2. 屏幕方向和布局变体：选择屏幕方向（横向和纵向），或选择应用提供备用布局的其他屏幕模式（例如夜间模式）。该菜单还包含用于创建新布局变体的命令；
3. 设备类型和尺寸：选择设备类型（手机/平板电脑、Android TV 或 Wear OS）和屏幕配置（尺寸和密度）。我们可以从多种预配置的设备类型和 AVD 定义中选择，也可以从列表中选择 Add Device Definition 创建新的 AVD；
4. API 版本：选择预览布局时使用的 Android 版本；
5. AppTheme：选择要应用于预览的界面主题背景。请注意，这仅适用于支持的布局样式，因此该列表中的许多主题背景都会导致出错；
6. Language：选择要以何种语言显示界面字符串。此列表仅会显示我们的字符串资源支持的语言。如果想要修改翻译，点击点击下拉菜单中的 Edit Translations。

如何修改视图属性

查看布局的代码（Code），修改Textview的Text属性

```
    android:text="Hello World!"
```

修改字符串属性值为“Hello Kotlin!”。更进一步，修改字体显示属性，在Design视图中选择textview_first文本组件，在Common Attributes属性下的 textAppearance域，设置相关的文字显示属性



查看布局的XML代码，可以看到新属性被应用。

```
android:fontFamily="sans-serif-condensed"
android:text="@string/hello_kotlin"
android:textColor="#9C27B0"
android:textSize="24sp"
```

4、向页面添加更多的布局

本步骤将向第一个Fragment添加更多的视图组件

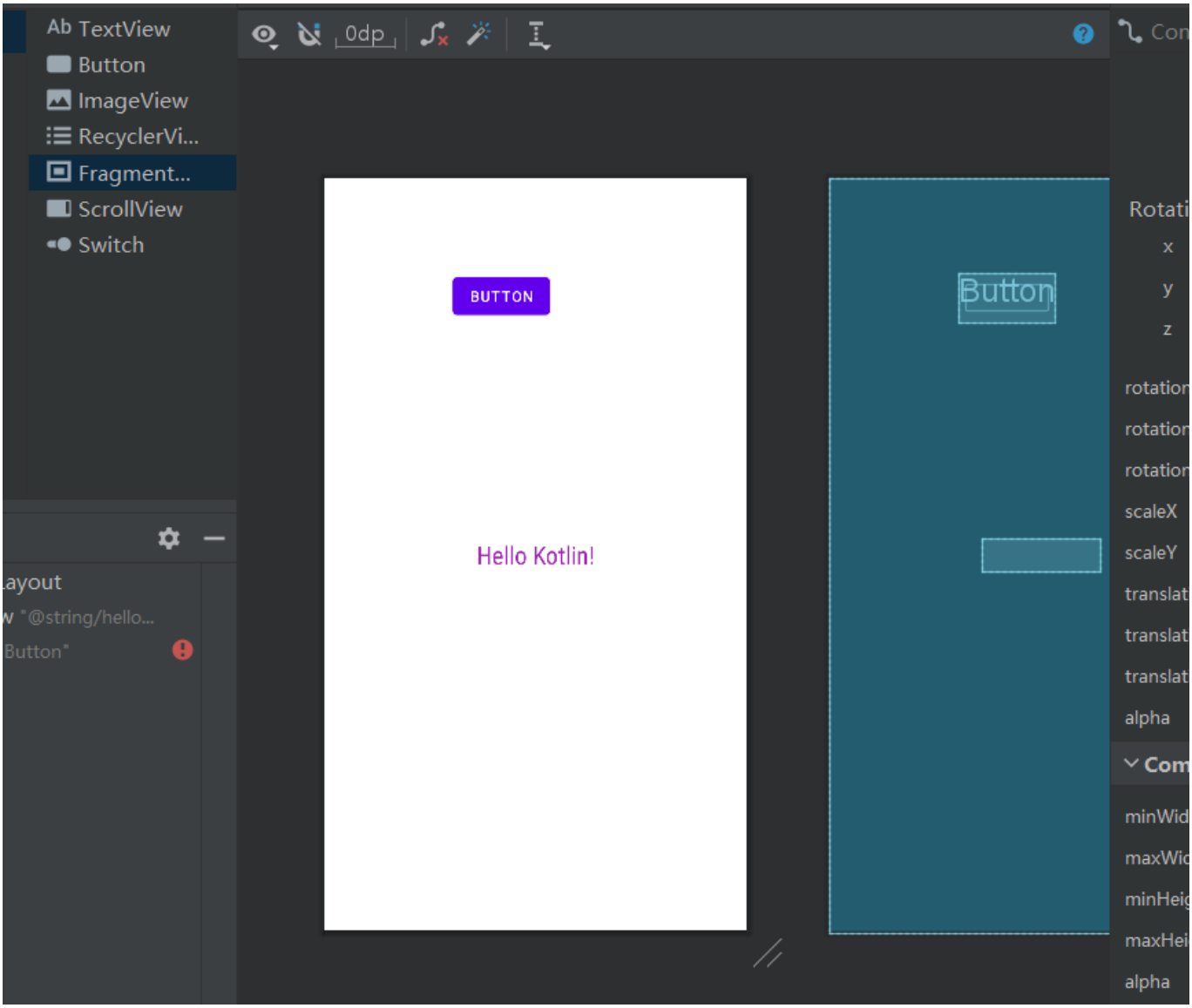
查看视图的布局约束

在fragment_first.xml，查看TextView组件的约束属性在约束布局中，每个组件至少需要一个水平方向和一个垂直方向的约束：

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

添加按钮和约束

从Palette面板中拖动Button

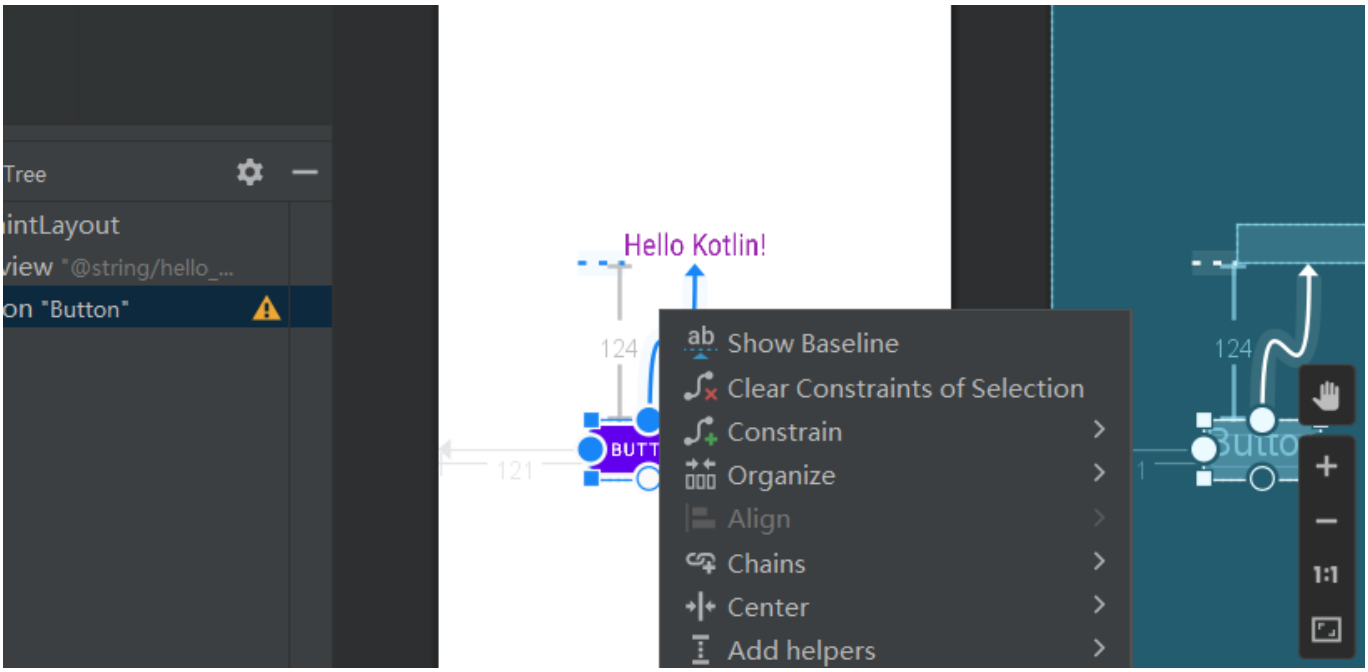


调整Button的约束，设置Button的Top>BottonOf textView

```
app:layout_constraintTop_toBottomOf="@+id/textview"
```

调整按钮

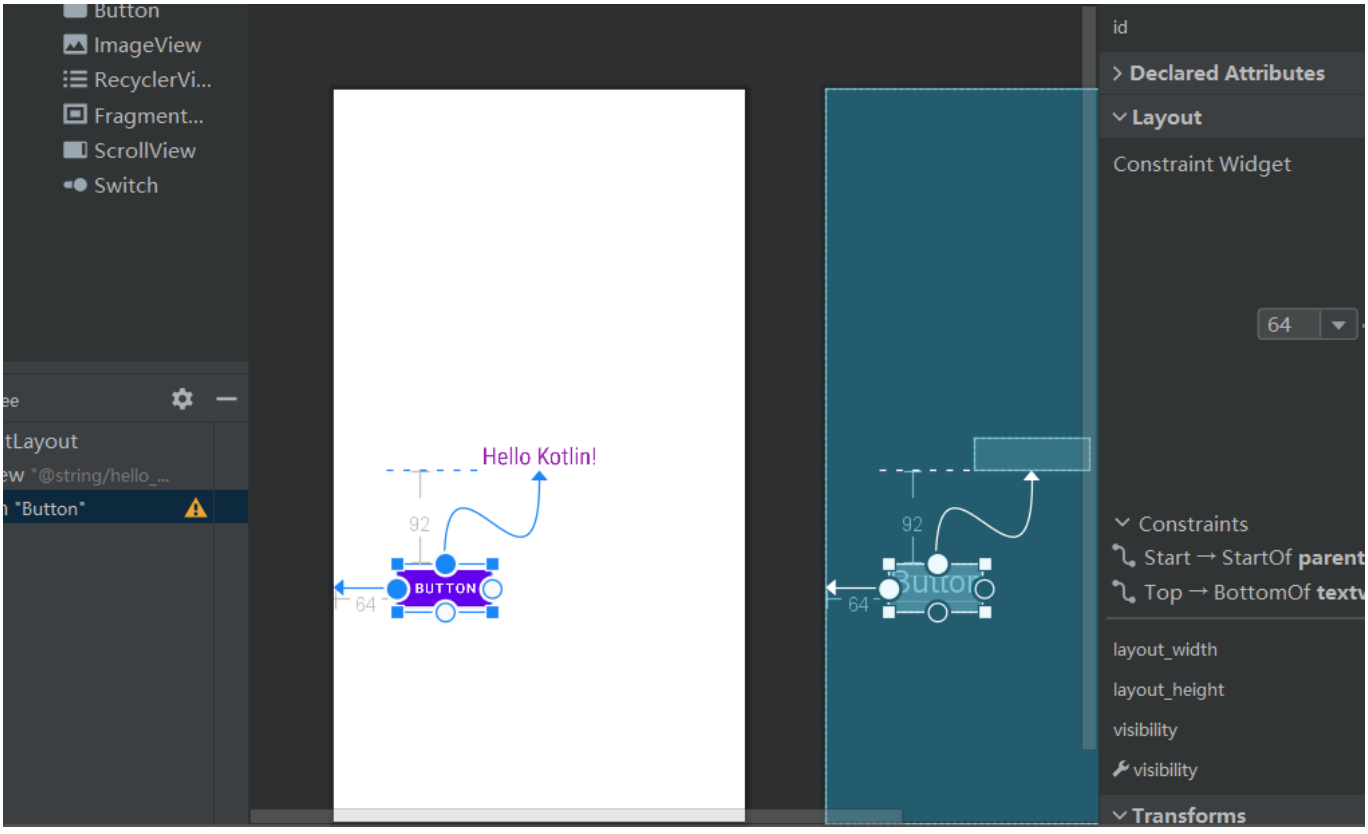
查看按钮的布局设计视图，它与TextView之间的连接不是锯齿状的而是波浪状的，表明两者之间存在链（chain），是一种两个组件之间的双向联系而不是单向联系。删除两者之间的链，可以在设计视图右键相应约束，选择Delete



同时，删除按钮的左侧约束。

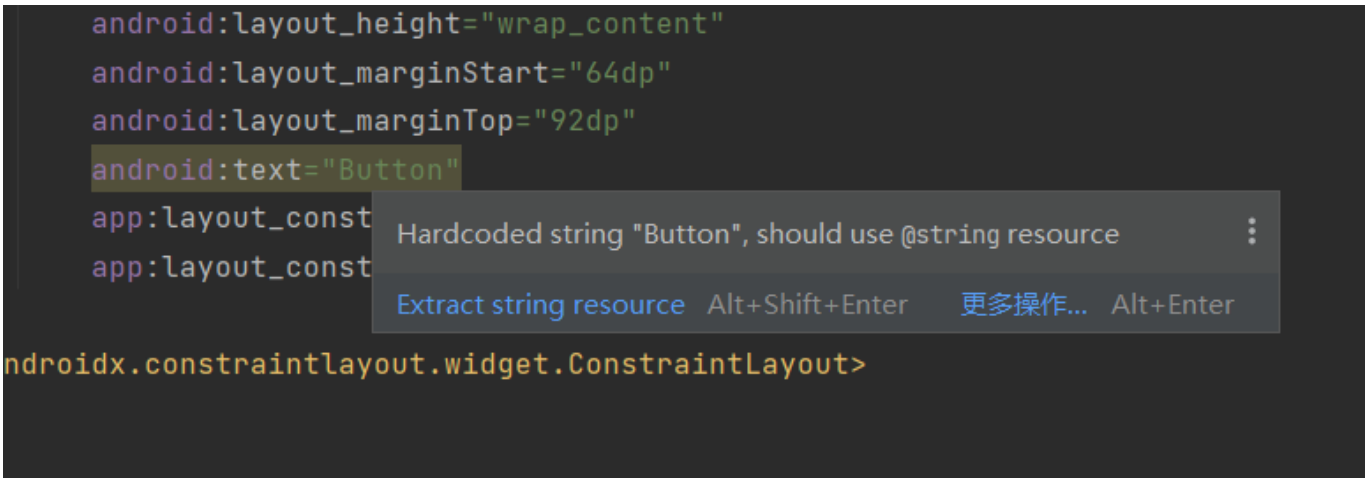
添加新的约束

添加按钮的右边和底部约束至父类屏幕，Top约束至TextView的底部，效果如下

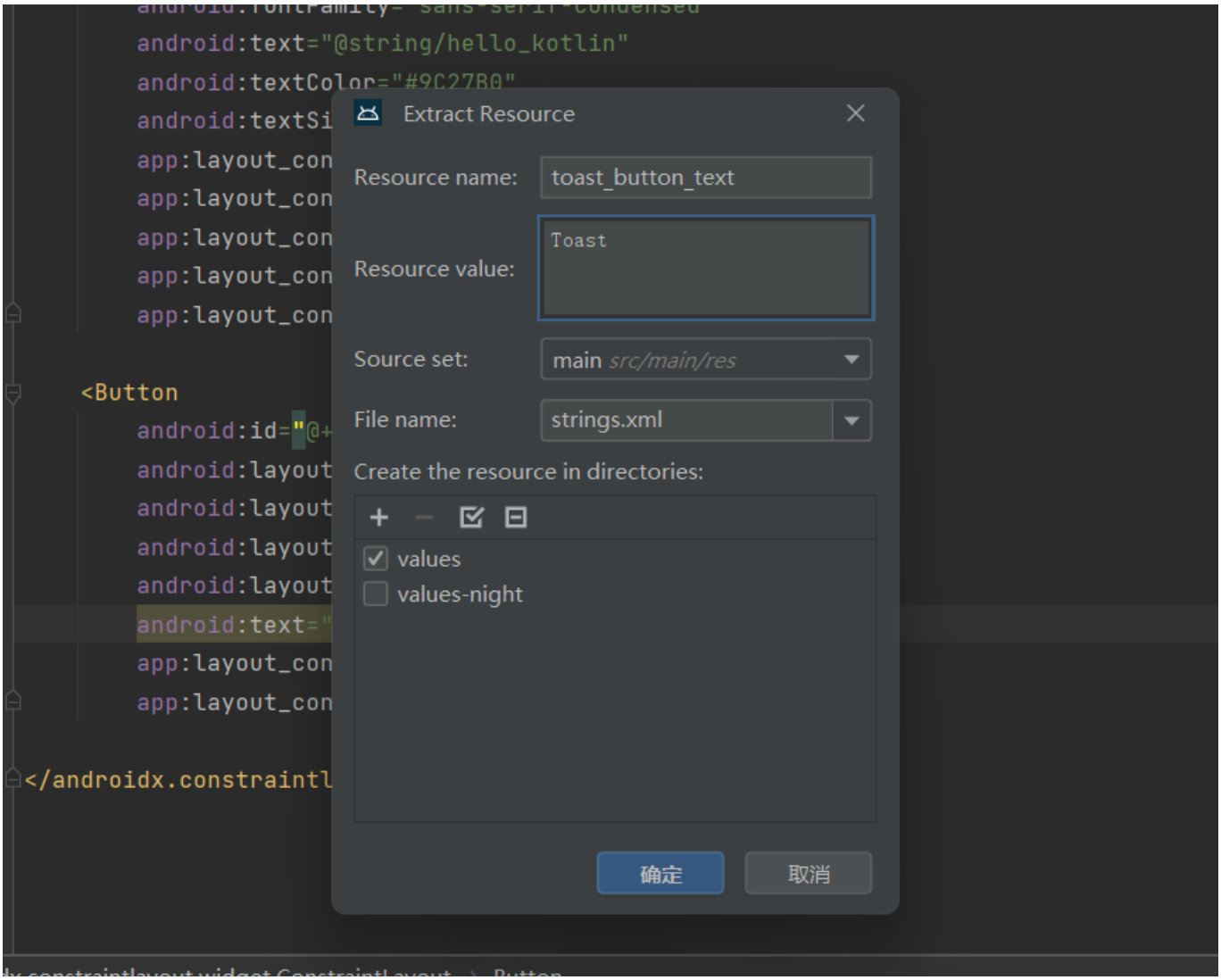


更改组件的文本

在布局文件代码中，找到按钮的text属性部分，点击文本，左侧出现灯泡状的提示，选择 Extract string resource。

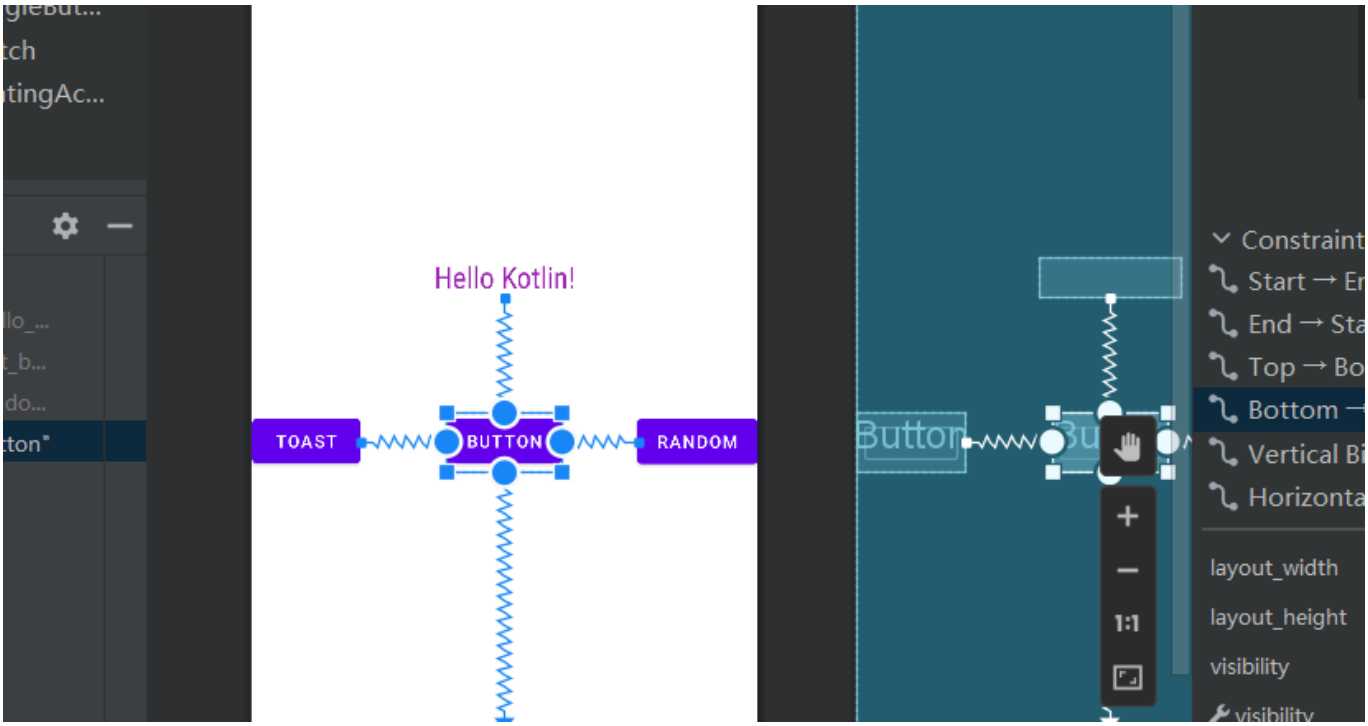


弹出对话框，令资源名为toast_button_text，资源值为Toast，并点击OK。



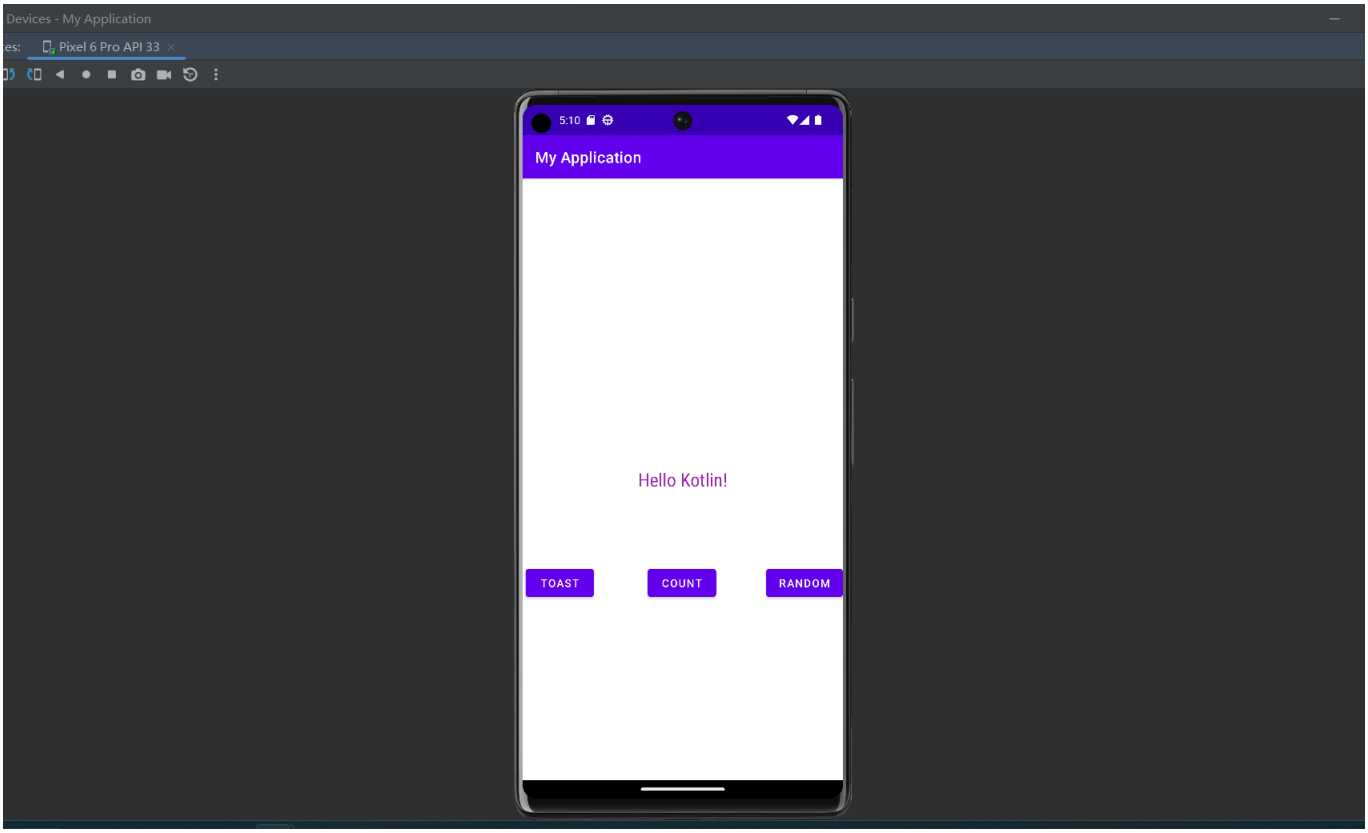
添加新的按钮

向fragment_first.xml文件中添加第三个按钮，位于Toast和Random按钮之间，TextView的下方。新Button的左右约束分别约束至Toast和Random，Top约束至TextView的底部，Buttom约束至屏幕的底部，效果如下：



运行应用程序查看效果

效果如下：



5、更新按钮和文本框的外观

添加新的颜色资源

values>colors.xml定义了一些应用程序可以使用的颜色，添加新颜色screenBackground 值为 #2196F3，这是蓝色阴影色；添加新颜色buttonBackground 值为 #BBDEFB

```
<color name="screenBackground">#2196F3</color>
<color name="buttonBackground">#BBDEFB</color>
```

设置组件的外观

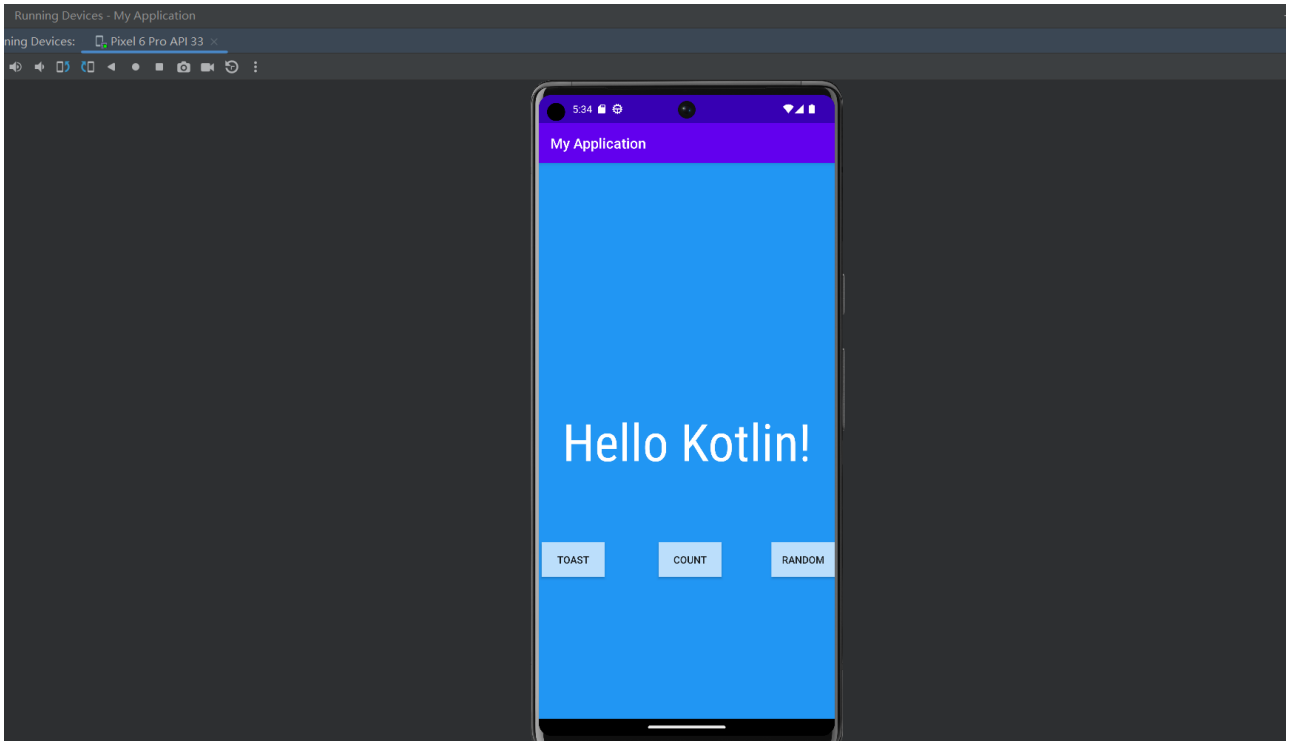
1. 属性面板中设置屏幕背景色为

```
android:background="@color/screenBackground"
```

2. 设置每个按钮的背景色为buttonBackground

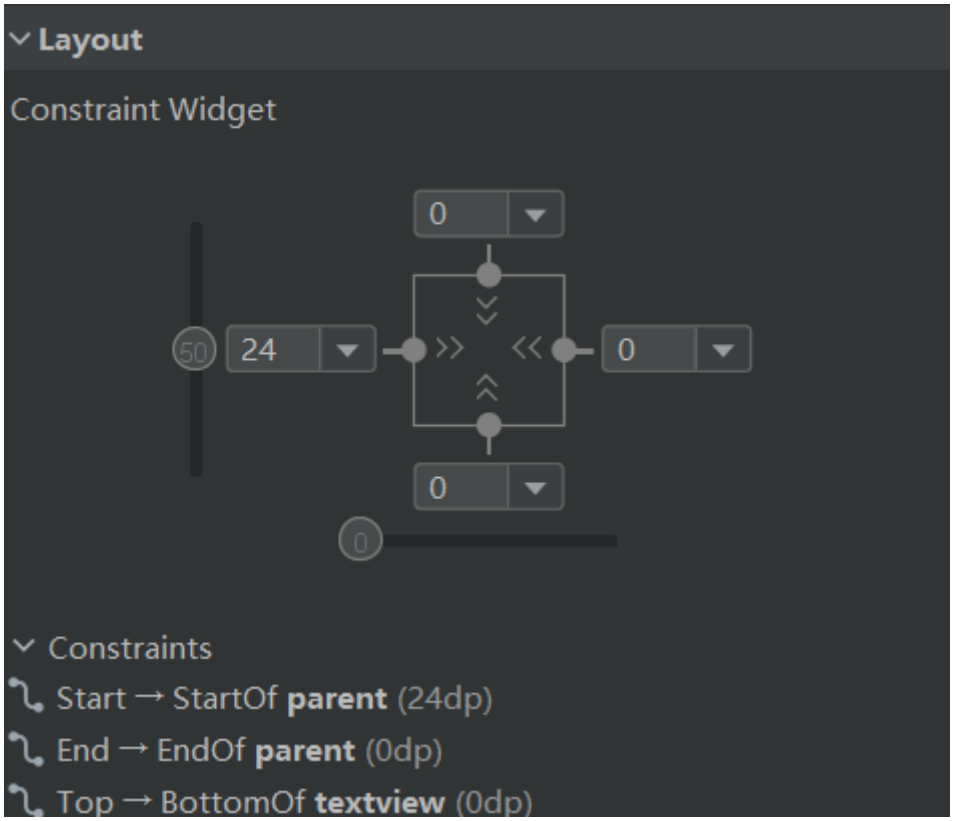
```
android:background="@color/buttonBackground"
```

3. 移除TextView的背景颜色，设置TextView的文本颜色为color/white，并增大字体大小至72sp
4. 效果:



设置组件的位置

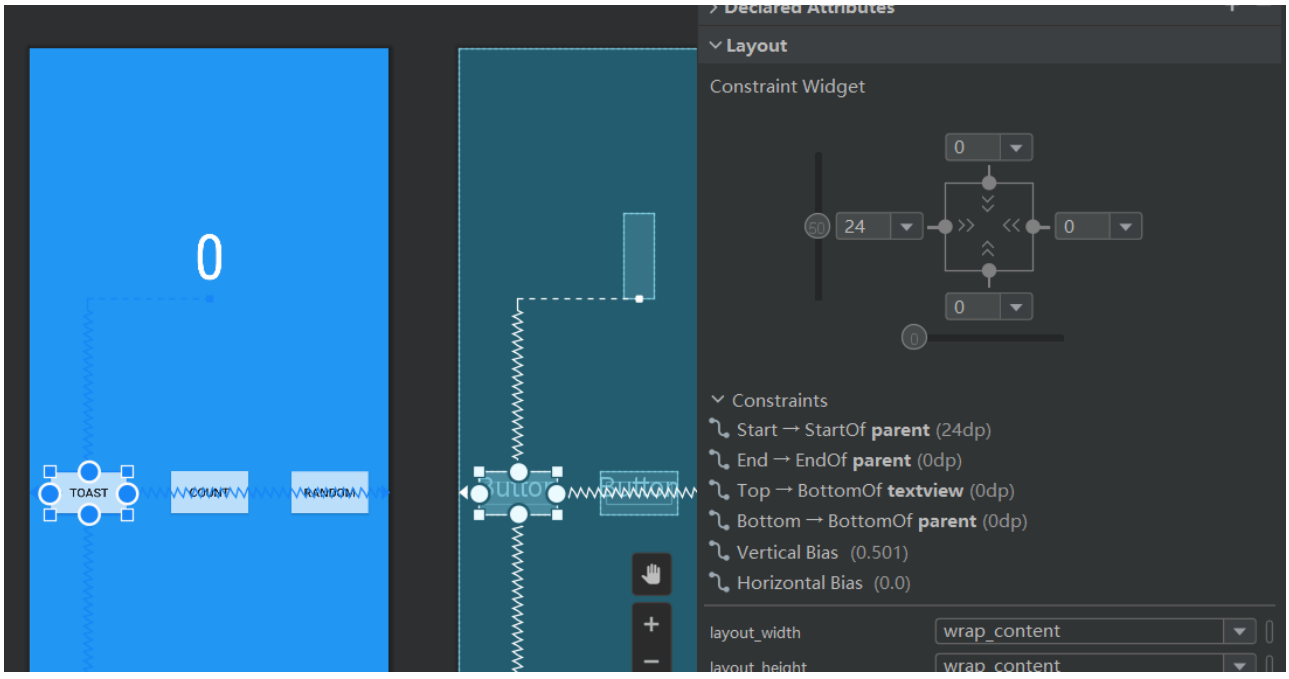
1. Toast与屏幕的左边距设置为24dp，Random与屏幕的右边距设置为24dp，利用属性面板的Constraint Widget完成设置。



2. 设置TextView的垂直偏移为0.3

```
app:layout_constraintVertical_bias="0.3"
```

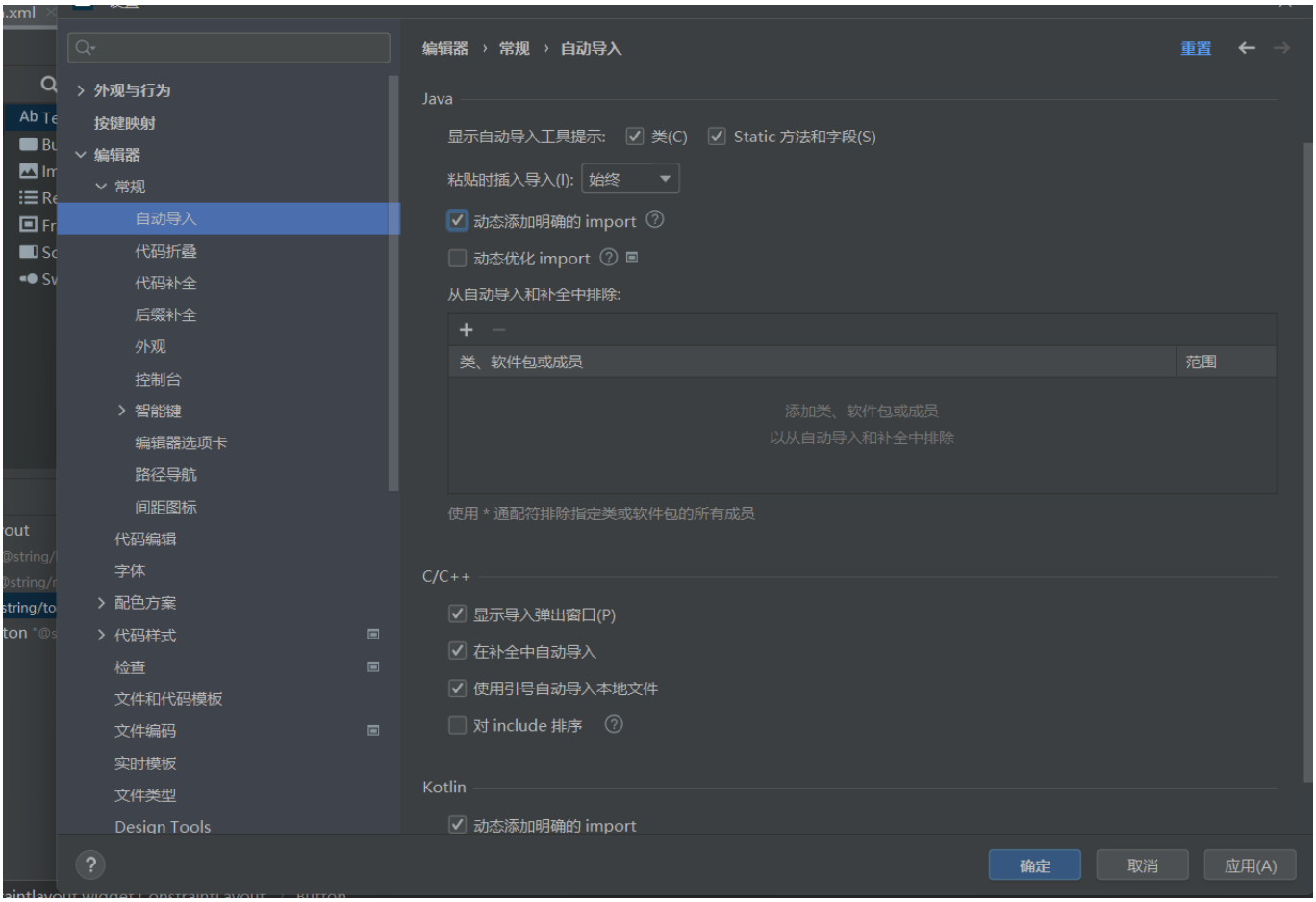
3. 效果:



6、添加代码完成应用程序交互

设置代码自动补全

Android Studio中，依次点击File>New Projects Settings>Settings for New Projects...，查找Auto Import选项，在Java和Kotlin部分，勾选Add Unambiguous Imports on the fly。

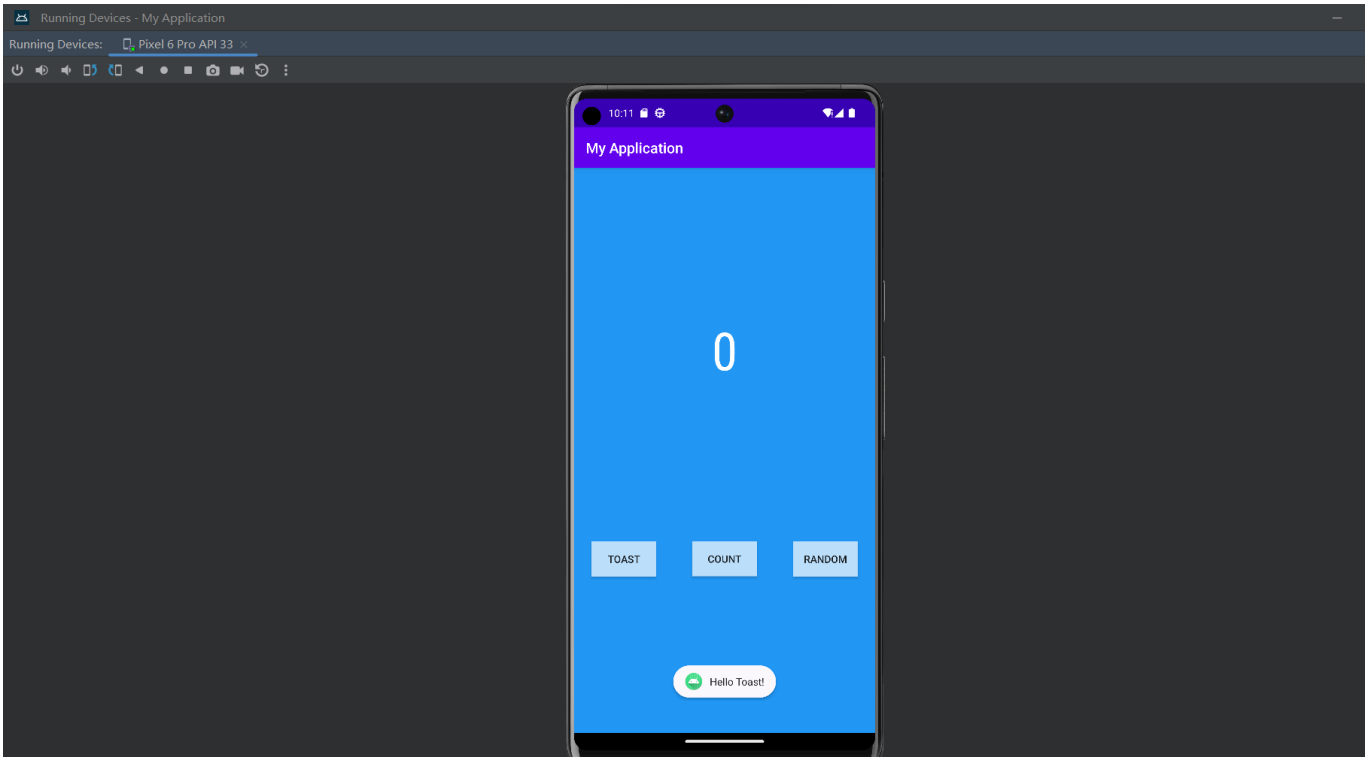


按钮添加一个toast消息

Toast 是Android 系统提供的一种非常好的提醒方式，在程序中可以使用它将一些短小的信息通知给用户，这些信息会在一段时间后自动消失，并且不会占用任何屏幕空间。在onCreate()方法中添加如下代码：

```
 // find the toast_button by its ID and set a click listener
findViewById<Button>(R.id.toast_button).setOnClickListener {
    // create a Toast with some text, to appear for a short time
    val myToast = Toast.makeText(this, "Hello Toast!", Toast.LENGTH_LONG)
    // show the Toast
    myToast.show()
}
```

效果如下：



使Count按钮更新屏幕的数字

此步骤向Count按钮添加事件响应，更新Textview的文本显示。

```
findViewById<Button>(R.id.count_button).setOnClickListener {
        countMe()
}
```

countMe()为自定义方法，每次点击增加数字1，具体代码为：

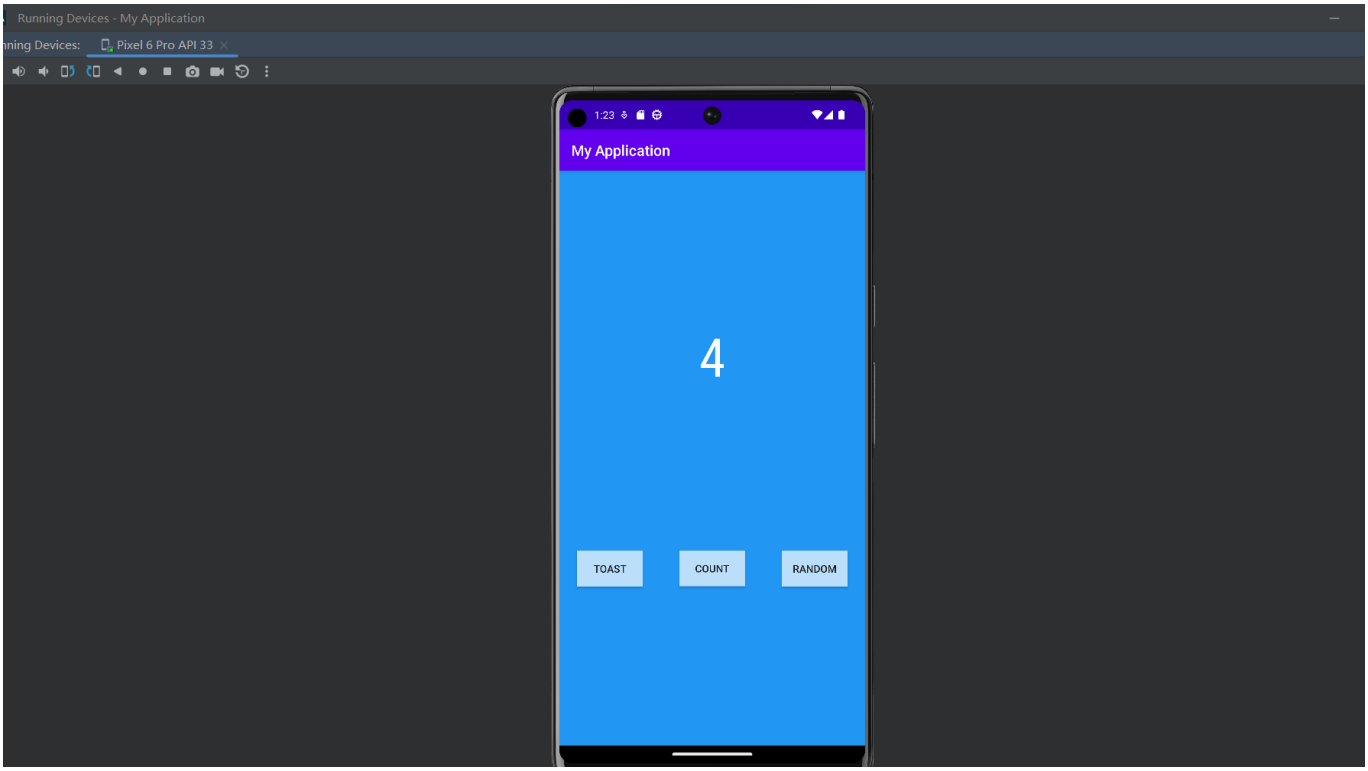
```
  private fun countMe() {
    // Get the text view
    val showCountTextView = findViewById<TextView>(R.id.textview)

    // Get the value of the text view.
    val countString = showCountTextView.text.toString()

    // Convert value to a number and increment it
    var count = countString.toInt()
    count++

    // Display the new value in the text view.
    showCountTextView.text = count.toString()
}
```

效果如下：



7、完成第二界面的代码

向界面添加TextView显示随机数

1. 打开fragment_second.xml的设计视图中，当前界面有两个组件，一个Button和一个TextView（textview_second）。
2. 去掉TextView和Button之间的约束
3. 拖动新的TextView至屏幕的中间位置，用来显示随机数
4. 设置新的TextView的id为**@+id/textview_random**
5. 设置新的TextView的左右约束至屏幕的左右侧，Top约束至textview_second的Bottom，Bottom约束至Button的Top
6. 设置TextView的字体颜色textColor属性为**@android:color/white**，textSize为72sp，textStyle为bold
7. 设置TextView的显示文字为“R”
8. 设置垂直偏移量layout_constraintVertical_bias为0.45 新增TextView最终的属性代码：

```
<TextView
    android:id="@+id/textview_random"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="R"
    android:textColor="@android:color/white"
    android:textSize="72sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/button_second"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textview_second"
    app:layout_constraintVertical_bias="0.45" />
```

更新显示界面文本的TextView

1. 更改该文本框id为textview_header
2. 设置layout_width为match_parent，layout_height为wrap_content。
3. 设置top，left和right的margin为24dp，左边距和右边距也就是start和end边距。
4. 若还存在与Button的约束，则删除。
5. 向colors.xml添加颜色colorPrimaryDark，并将TextView颜色设置为@color/colorPrimaryDark，字体大小为24sp。
6. strings.xml文件中，修改hello_second_fragment的值为"Here is a random number between 0 and %d."
7. 使用Refactor>Rename将hello_second_fragment 重构为random_heading 新增TextView最终的属性代码：

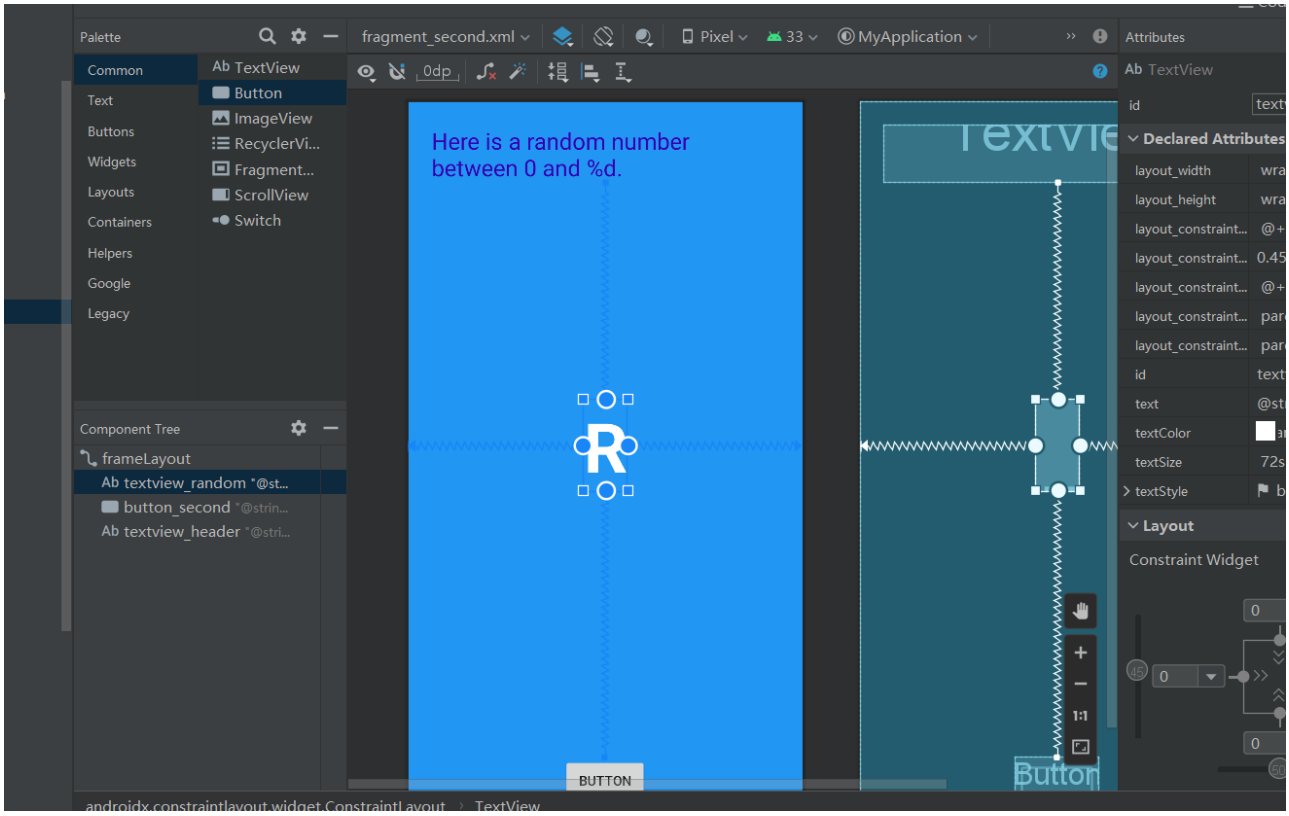
```
<TextView
   android:id="@+id/textview_header"
   android:layout_width="0dp"
   android:layout_height="wrap_content"
   android:layout_marginStart="24dp"
   android:layout_marginLeft="24dp"
   android:layout_marginTop="24dp"
   android:layout_marginEnd="24dp"
   android:layout_marginRight="24dp"
   android:text="@string/random_heading"
   android:textColor="@color/colorPrimaryDark"
   android:textSize="24sp"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toTopOf="parent" />
```

更改界面的背景色和按钮布局

1. 向colors.xml文件添加第二个Fragment背景色的值，修改fragment_second.xml背景色的属性为screenBackground2

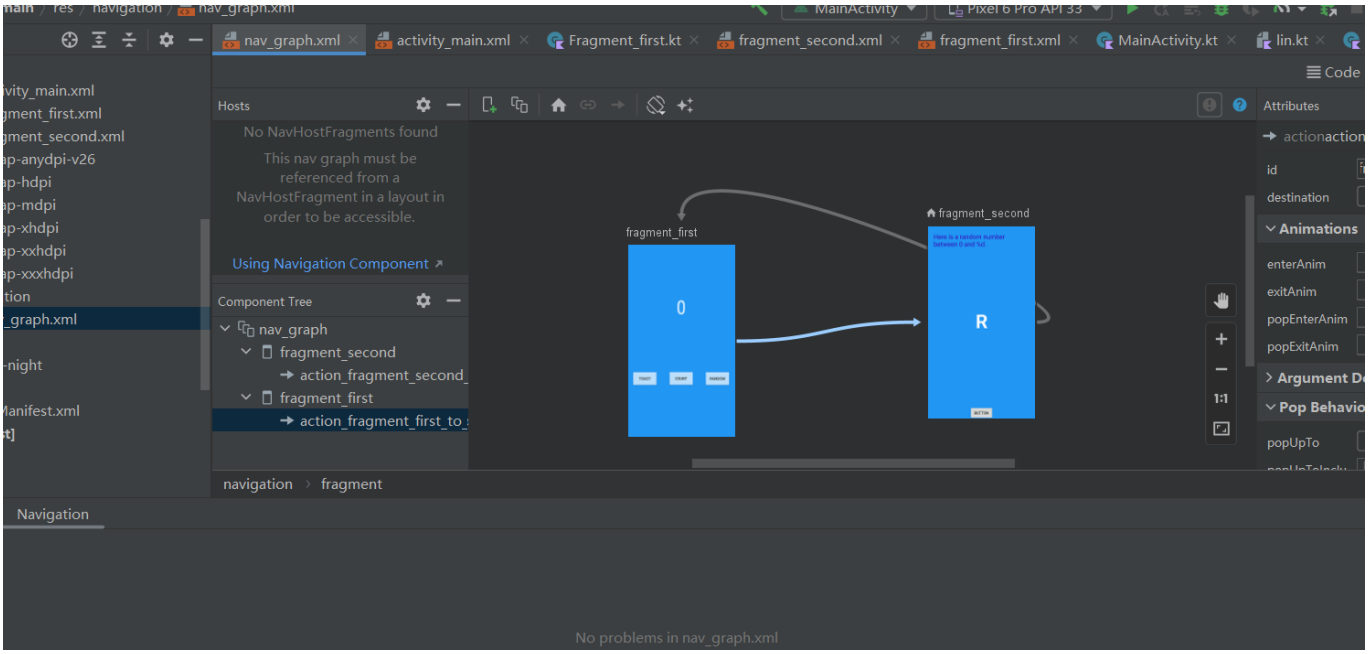
```
<color name="screenBackground2">#26C6DA</color>
```

2. 将按钮移动至界面的底部，完成所有布局之后，如下图所示：



检查导航图

检查导航图:



启用SafeArgs

SafeArgs 是一个 gradle 插件，它可以帮助您在导航图中输入需要传递的数据信息，作用类似于Activity之间传递数据的Bundle。

1. 首先打开 Gradle Scripts > build.gradle(Project: My First App)
2. 找到buildscript脚本中的dependencies章节，添加如下代码

```
def nav_version = "2.3.0-alpha02"
classpath "androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version"
```

3. 接着打开 Gradle Scripts > build.gradle (Module: app)
4. apply plugin开头的代码下添加一行

```
apply plugin: 'androidx.navigation.safeargs.kotlin'
```

创建导航动作的参数

1. 打开导航视图，点击FirstFragment，查看其属性。
2. 在Actions栏中可以看到导航至SecondFragment
3. 同理，查看SecondFragment的属性栏
4. 点击Arguments **+**符号
5. 弹出的对话框中，添加参数myArg，类型为整型Integer

FirstFragment添加代码，向SecondFragment发数据

初始应用中，点击FirstFragment的Next/Random按钮将跳转到第二个页面，但没有传递数据。在本步骤中将获取当前TextView中显示的数字并传输至SecondFragment。

1. 打开FirstFragment.kt源代码文件
2. 找到onViewCreated()方法，该方法在onCreateView方法之后被调用，可以实现组件的初始化。找到Random按钮的响应代码，注释掉原先的事件处理代码

```
val showCountTextView = view.findViewById<TextView>(R.id.textview_first)
val currentCount = showCountTextView.text.toString().toInt()
```

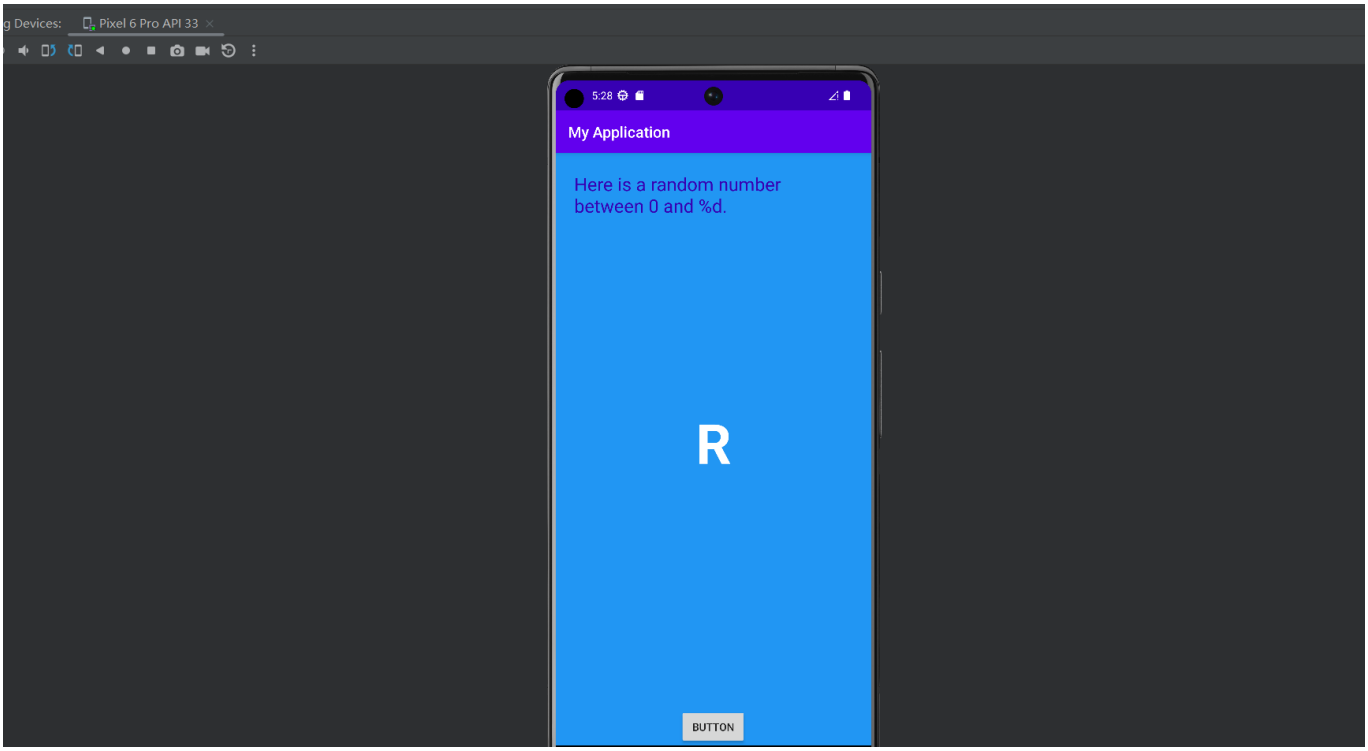
4. 将currentCount传递给actionFirstFragmentToSecondFragment ()

```
val action =
FirstFragmentDirections.actionFirstFragmentToSecondFragment(currentCount)
```

5. 添加导航事件代码

```
   findNavController().navigate(action)
```

运行代码，点击FirstFragment的Count按钮，然后点击Random按钮，可以看到屏幕中间还未出现随机数显示:



添加SecondFragment的代码

1. onViewCreated()代码之前添加一行

```
val args: SecondFragmentArgs by navArgs()
```

2. onViewCreated()中获取传递过来的参数列表，提取count数值，并在textview_header中显示

```
val count = args.myArg
val countText = getString(R.string.random_heading, count)
view.findViewById<TextView>(R.id.textview_header).text = countText
```

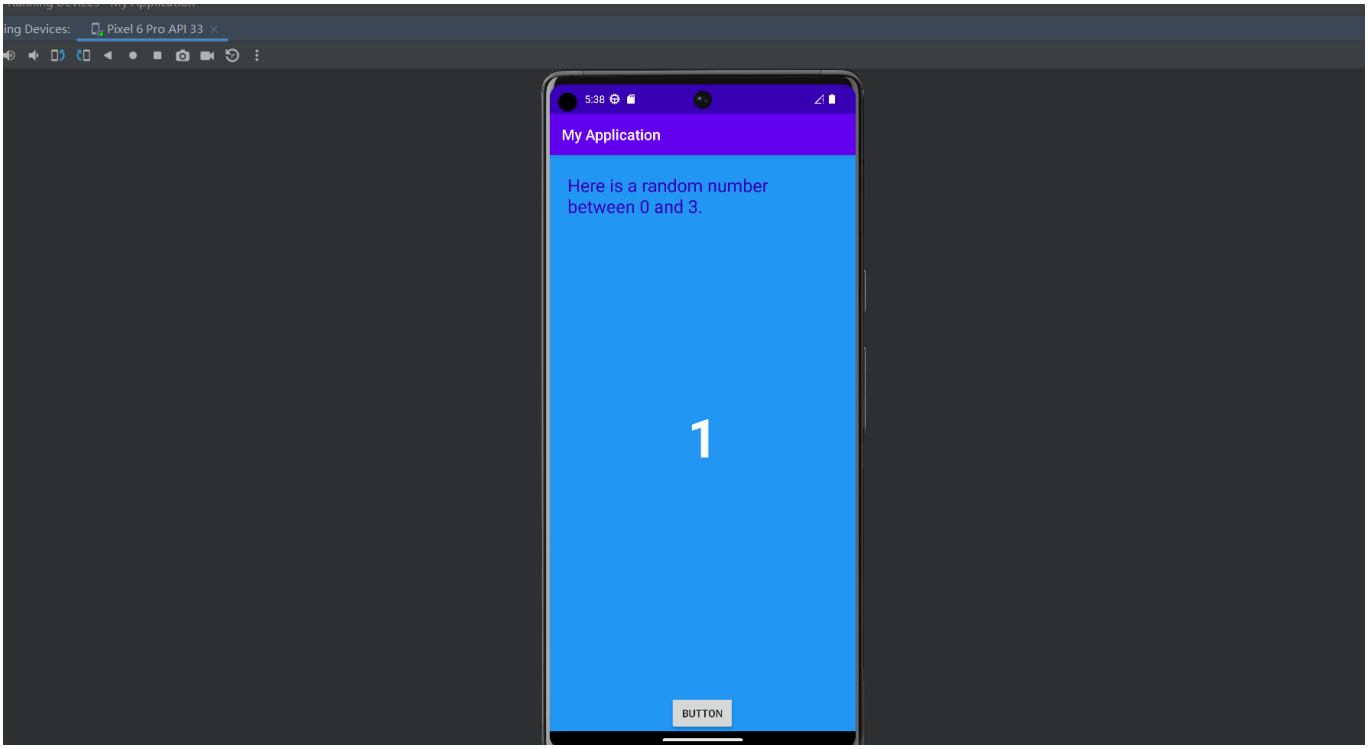
3. 根据count值生成随机数

```
val random = java.util.Random()
var randomNumber = 0
if (count > 0) {
   randomNumber = random.nextInt(count + 1)
}
```

4. extview_random中显示count值

```
view.findViewById<TextView>(R.id.textview_random).text = randomNumber.toString()
```

最终运行结果


ing Devices:
Pixel 6 Pro API 33
5:38
My Application
Here is a random number
between 0 and 3.
1
BUTTON