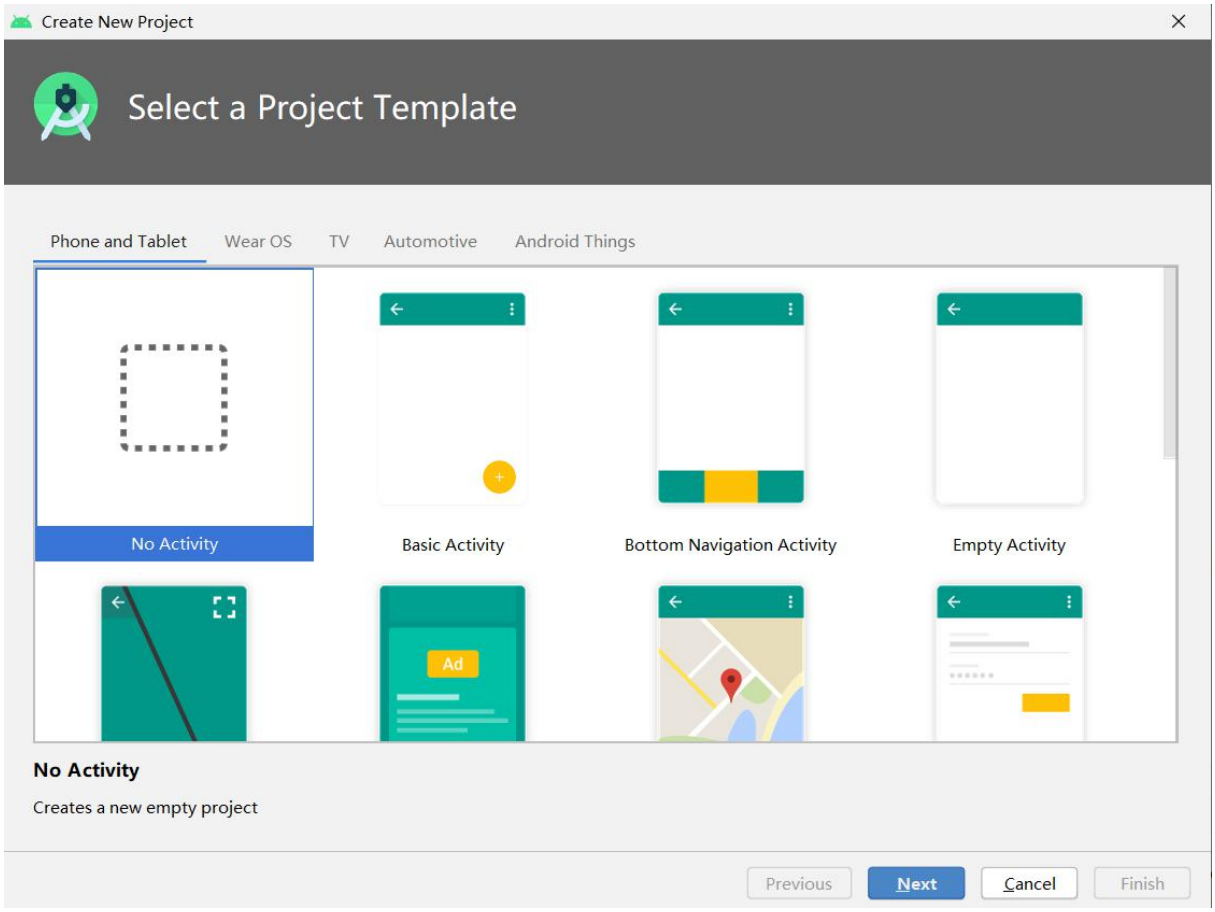
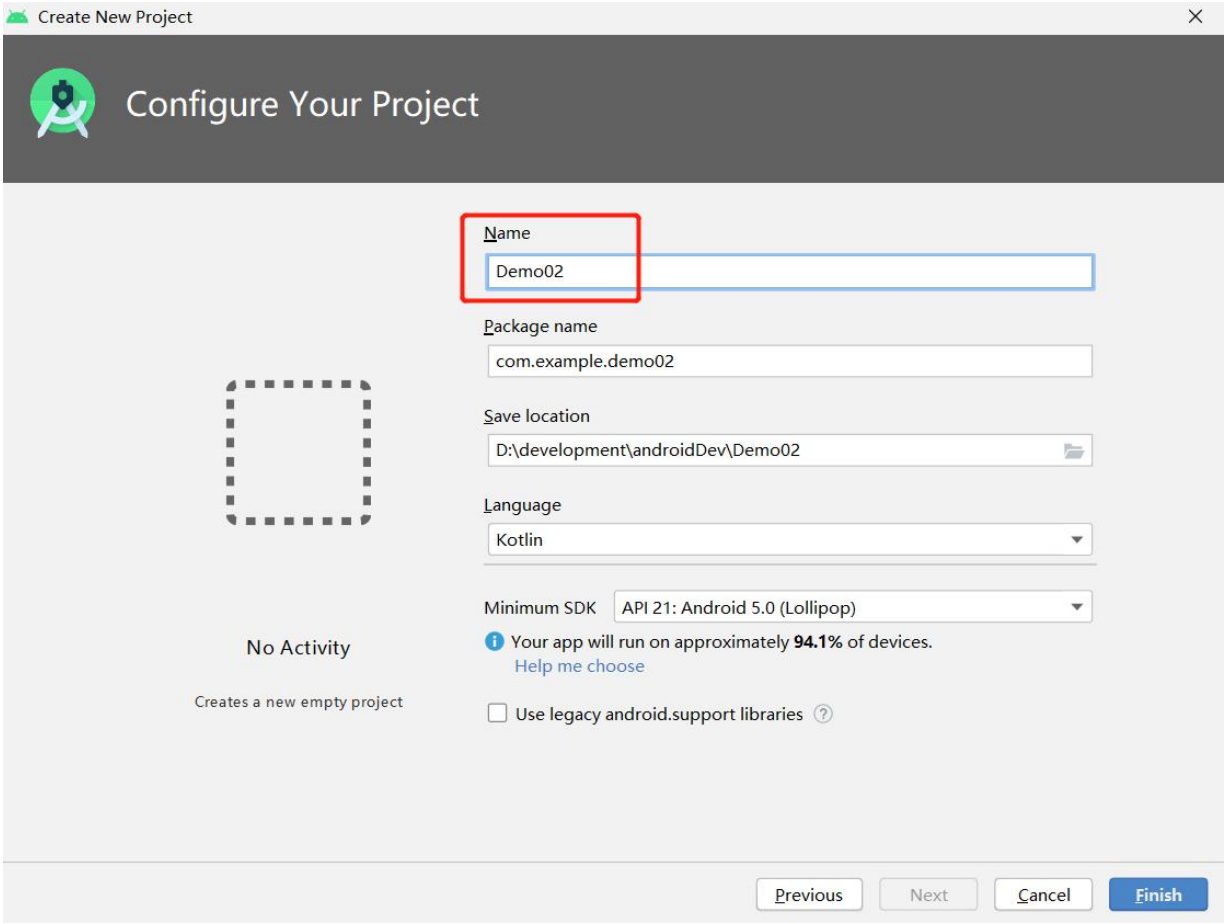
Activity 的最佳实践

1、新建项目

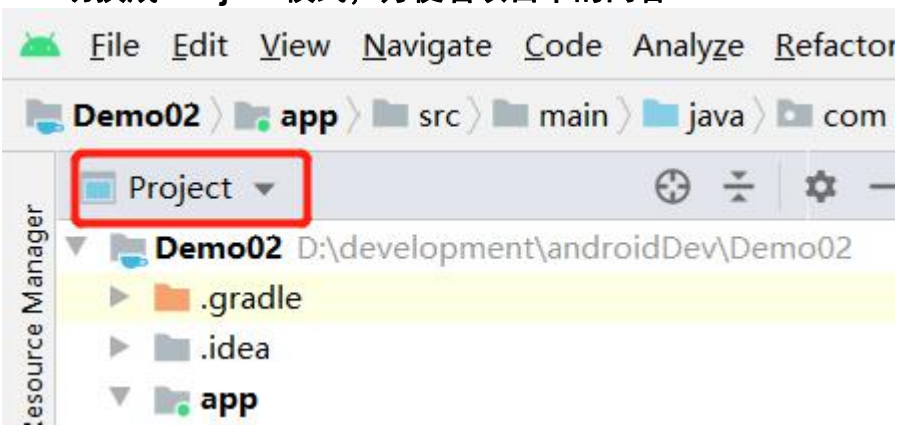
请参照上次实验内容新建项目，项目名称为 Demo02，请注意，这次选择 Add no Activity





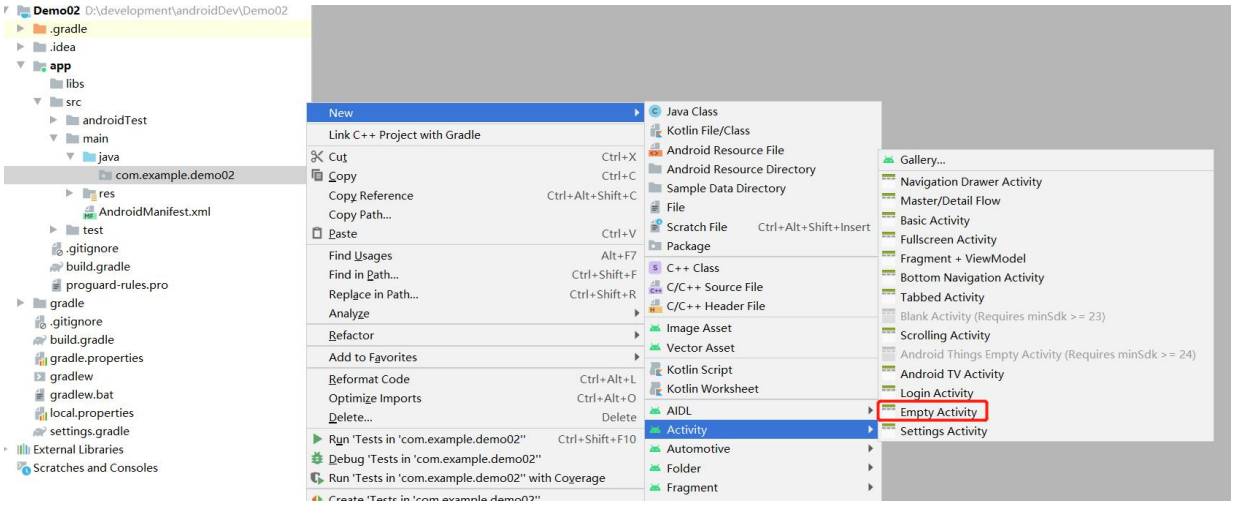
2、手动创建 Activity

2.1 切换成 Project 模式，方便看项目中的内容

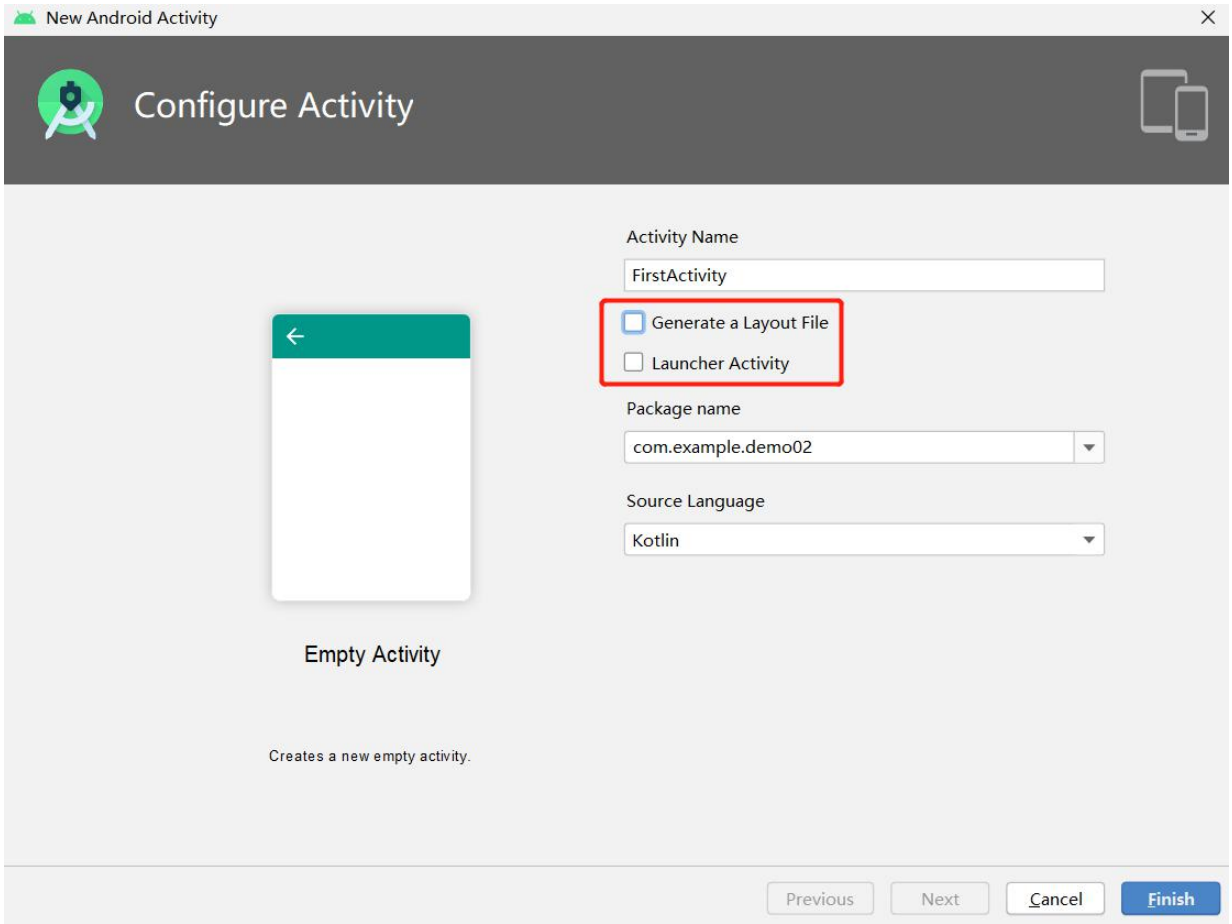


2.2 手动创建一个 Activity

将手动创建的 Activity 起名为 FirstActivity



不要勾选自动生成 Layout 及运行 Activity 两个选项

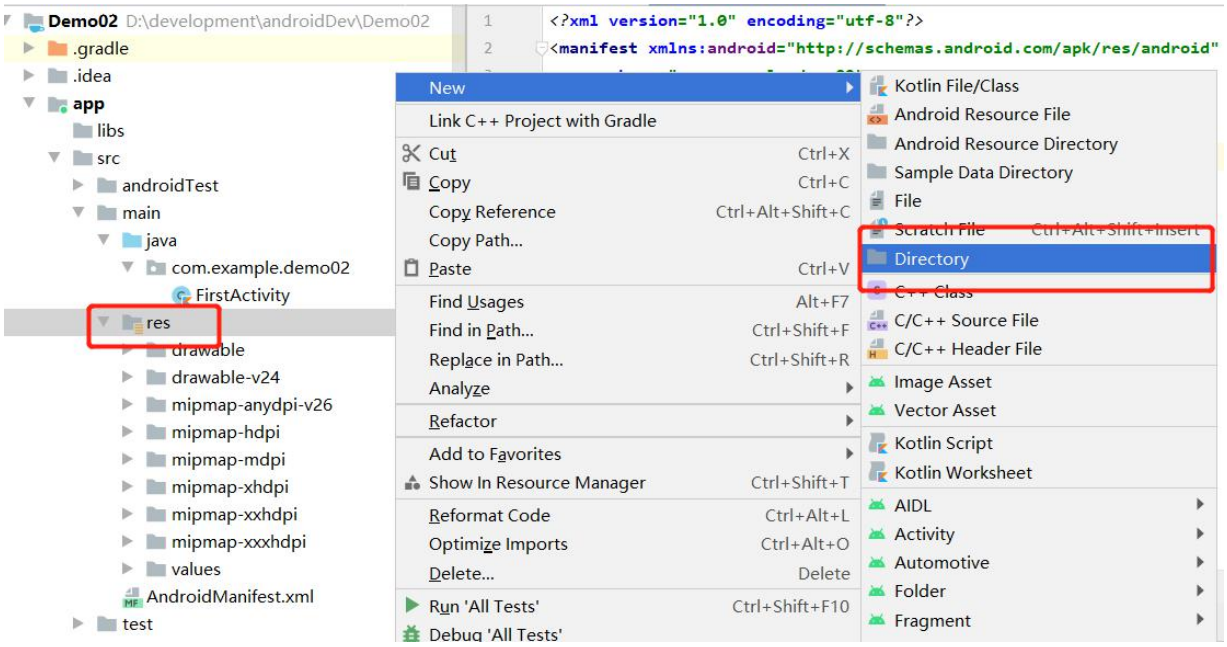


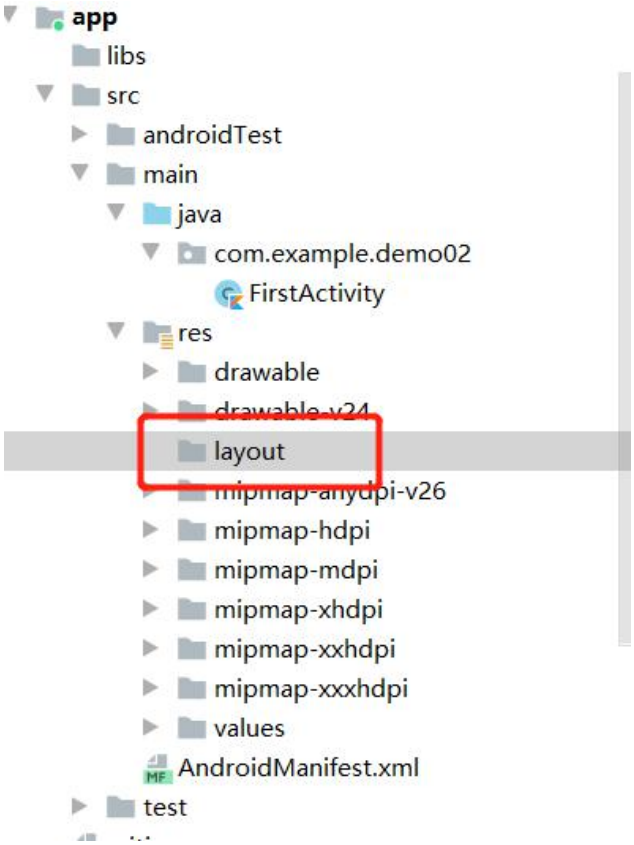
创建完毕后，你会看到，这样创建出来的 Activity 是没有布局文件，同时描述文件中也没有启动信息

2.3 创建和加载布局

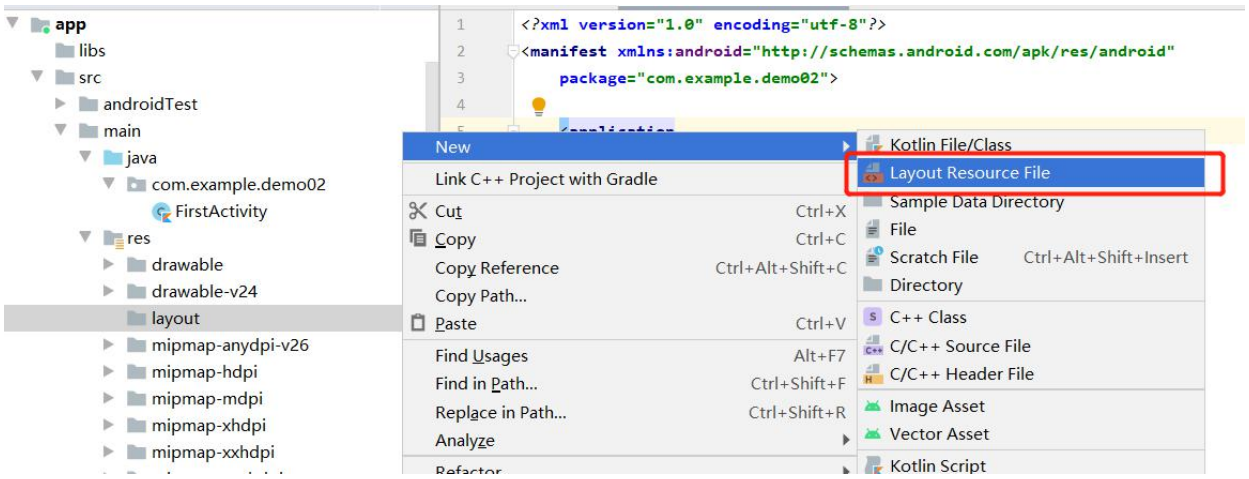
Android 的设计讲究逻辑和视图分离，上一步构建的是逻辑，这一步要构建出视图，一个 Activity 对应于一个 Layout。

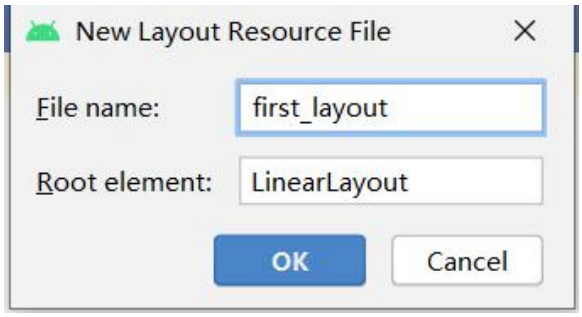
先新建视图文件所在的文件夹（如果不存在的话）



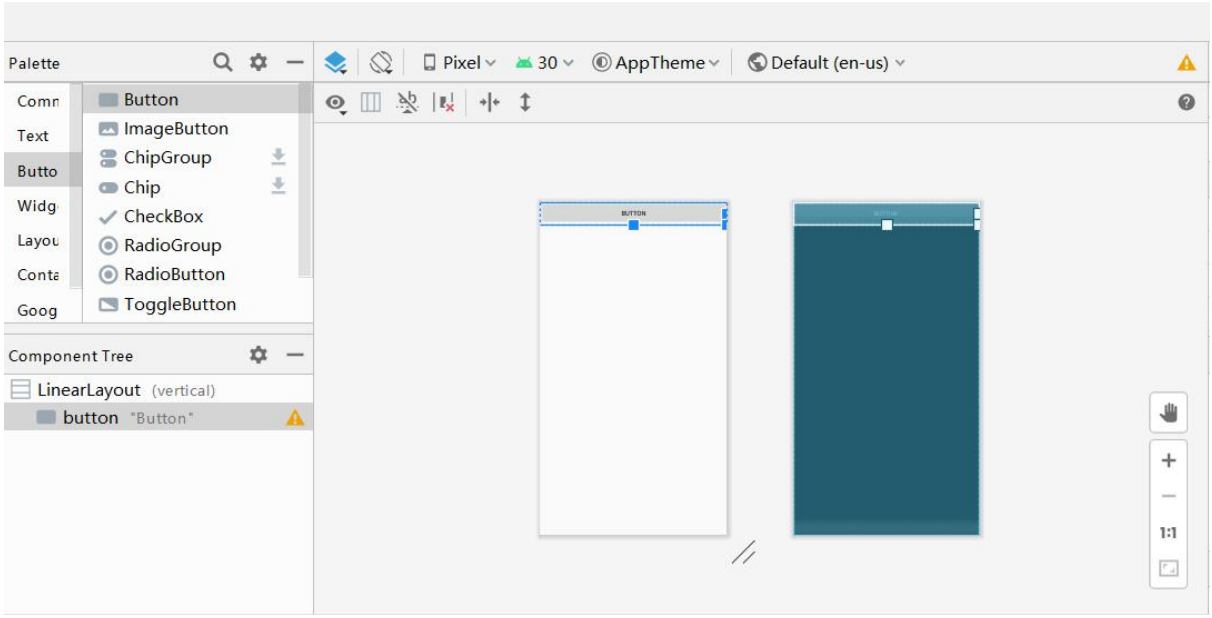


创建布局文件





打开新建的布局文件，放入一个按钮



将按钮上的文本改成 android:text="open Second activity"

2.4 注册启动 Activity 到描述文件中

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.demo02">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Demo02"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".FirstActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2.5 将布局文件注册到 Activity 中

```kotlin
class FirstActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.first_layout)
    }
}
```

此处目的是为了将布局文件与 Activity 关联。

2.6 最后

运行程序，看是否能正常启动

3、创建其他 Activity

如第二步所示，创建 SecondActivity 及其布局文件，创建 ThirdActivity 及其布局文件，上面各放置按钮一枚

注意，此处无需再注册启动信息。

4、从 FirstActivity 打开 SecondActivity

因为控件还没有学，这里我先给出代码

4.1 FirstActivit 对应的布局文件中注册按钮的点击事件

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

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button"
        android:onClick="click"/>
</LinearLayout>
```

4.2 FirstActivity 中注册该事件

```
class FirstActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.first_layout)
    }

    fun click(view: View) {
        when (view.id){
            R.id.button -> {
                val intent = Intent( packageContext: this, SecondActivity::class.java)
                startActivity(intent)
            }
        }
    }
}
```

重新启动，测试是否能打开第二个 Activity

4.3 SecondActivity 放置两个按钮，第一个按钮同 FirstActivity，第二个按钮为“关闭”，注意重命名按钮 id

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

    <Button
        android:id="@+id/buttonOpen"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="open third activity"
        android:onClick="open"/>

    <Button
        android:id="@+id/buttonClose"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="close"
        android:onClick="close"/>
</LinearLayout>
```

```
class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.second_activity)
    }

    fun open(view: View) {
        val intent = Intent( packageContext: this, SecondActivity::class.java)
        startActivity(intent)
    }
    fun close(view: View) {
        finish()
    }
}
```

5、实现从 SecondActivity 跳转到 ThirdActivity

5.1 通过 second_layout 布局文件设置跳转页面的按钮：

```
<Button
    android:id="@+id/buttonOpen"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="open Third activity"
    android:onClick="open" />
```

5.2 SecondActivity 中，传递数据”Hello Third”到 ThirdActivity：

```
//打开页面3
fun open(view: View){
    val intent= Intent( packageContext: this,ThirdActivity::class.java)
    intent.putExtra( name: "data", value: "Hello Third");
    startActivity(intent)
}
```

5.3 ThirdActivity 中接收数据，调整 Toast 位置后用用它来显示

```
//显示页面2传递过来的消息
val data=intent.getStringExtra( name: "data")   //获取传递的内容
val toast=Toast.makeText( context: this, data, Toast.LENGTH_SHORT)
toast.setGravity(Gravity.CENTER_VERTICAL,  xOffset: 0,  yOffset: 0) //在屏幕中间显示
toast.show()
```

5.4 实现效果：

6、修改 SecondActivity 原代码，返回数据

6.1 修改 SecondActivity 的 close 函数，将数据”close second activity”放入 intent 中，并用 setResult 设置返回值

```
//关闭页面2
fun close(view:View){
    val intent2=Intent()
    intent2.putExtra( name: "data", value: "close second activity")
    setResult(Activity.RESULT_OK,intent2)     //设置返回值
    finish()
}
```

6.2 在 FirstActivity 中修改 startActivity 为 startActivityForResult 以便接收 SecondActivity 的返回值。

```
//打开页面2
fun click(view: View)
{
    when(view.id){
        R.id.button->{
            val intent = Intent( packageContext: this,SecondActivity::class.java)
            startActivityForResult(intent,  requestCode: 1);  //需要SecondActivity的返回值
        }
    }
}
```

6.3 然后用 override 重写 onActivityResult 方法，并用 Toast 显示数据

```
//重写onActivityResult方法
override fun onActivityResult(requestCode: Int,resultCode: Int,data: Intent?){
    super.onActivityResult(requestCode,resultCode,data)
    when(requestCode){
        1 -> if(resultCode == Activity.RESULT_OK){
            val returnedCode=data?.getStringExtra( name: "data")
            val toast=Toast.makeText( context: this,returnedCode,Toast.LENGTH_LONG)
            toast.setGravity(Gravity.CENTER_VERTICAL,  xOffset: 0,  yOffset: 0) //在屏幕中间显示
            toast.show()    //显示
        }
    }
}
```

6.4 效果：