

# Android基础

# 本章概要

---



Android 程序目录



Activity 基础



布局的添加与设置

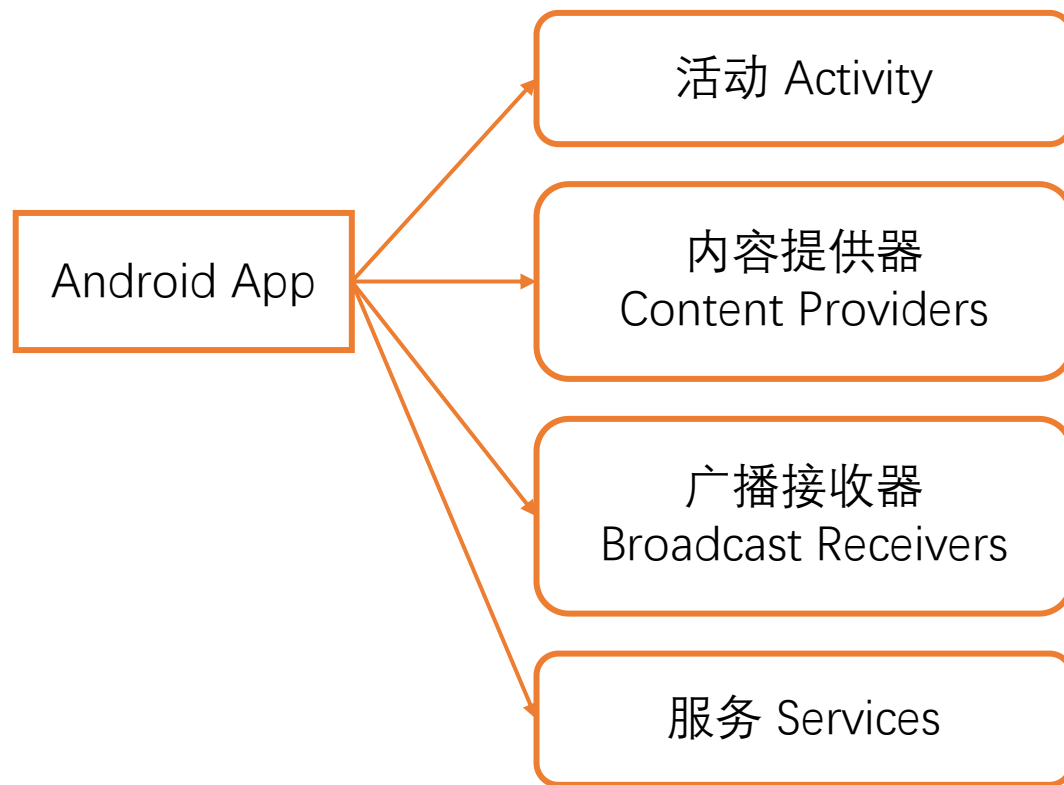


活动的生命周期

# Android 程序的组成

# Android 应用程序的组成组件

- ◆ Android 应用程序由多种组件 (Components) 组成, 这些组件是一些Android框架的Java类, 我们只需继承这些类, 重写和扩充其功能, 即可创建Android 应用程序。
- Android App组件分成四种 (但不一定每个APP都包含这4种组件)



# Android 应用程序的组成组件



## ◆活动 Activity

活动是Android应用程序与用户互动的组件，通俗地讲就是**用户界面**，有其自身的生命周期。

一般来说，一个活动就是用户在移动设备上看到的屏幕画面（大多是占满整个屏幕，不过也有可能是对话框）。一个Android App可以拥有多个活动，如同Web网站拥有多页网页。

# Android 应用程序的组成组件



## ◆内容提供者 Content Providers

内容提供程序是给Android App提供数据的程序。在Android中，很多时候App并不保存数据，数据由内容提供者进行管理。

例如：Android中的通讯录应用程序并没有储存任何通讯录数据，而是通过对应的Content Providers获取通讯录信息：用户姓名、地址、电话等。因此你也可以开发自己的通讯录程序，通过Content Providers访问和管理手机的通讯录信息。

内容提供者将数据的存储和数据的使用分割开来，提高了Android的弹性。

# Android 应用程序的组成组件

## Android App

活动 Activity

内容提供者  
Content Providers

广播接收器  
Broadcast Receivers

服务 Services

## ◆广播接收器 Broadcast Receivers

广播接收器用于接收系统广播并做出回应，是与Android系统进行交互的组件。例如，当手机接收到来电、收到短信、启用摄像头、电池剩余电量过低等等时，Android系统都会发出广播。

App中的广播接收器组件可以选择性的接受这些广播并做出回应。

同时，App也可以通过广播接收器来主动发布广播，与Android系统以及其他App进行信息交互。

# Android 应用程序的组成组件

## Android App

活动 Activity

内容提供者  
Content Providers

广播接收器  
Broadcast Receivers

服务 Services

## ◆服务 Services

服务是在幕后执行的进程，可以执行和活动一样的工作，只是不提供可视化的用户界面。比如音乐播放器在后台播放音乐就是通过服务的形式。

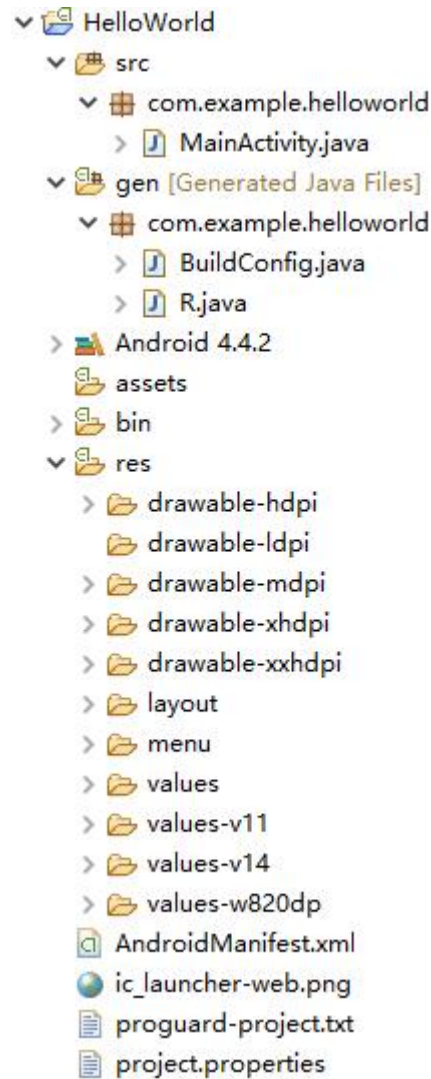
Android系统内置了很多系统服务，我们可以通过API来直接使用这些服务，例如定位服务。



# Android 应用程序工程结构

## ◆ 一个典型的Android App Project，它包含以下几个组成部分

- src文件夹：用户编写的Java程序
- gen文件夹：编译器生成的文件，其中R.java文件存放了资源的ID，**该文件内容不可以自行修改**
- res中是各种资源，包括图像、布局和字符串等，他们都有惟一个ID
- assets也可放资源，但不会产生ID（常用于放HTML文件、文本文件和SQLite数据库等）
- AndroidManifest.xml是App的配置文件

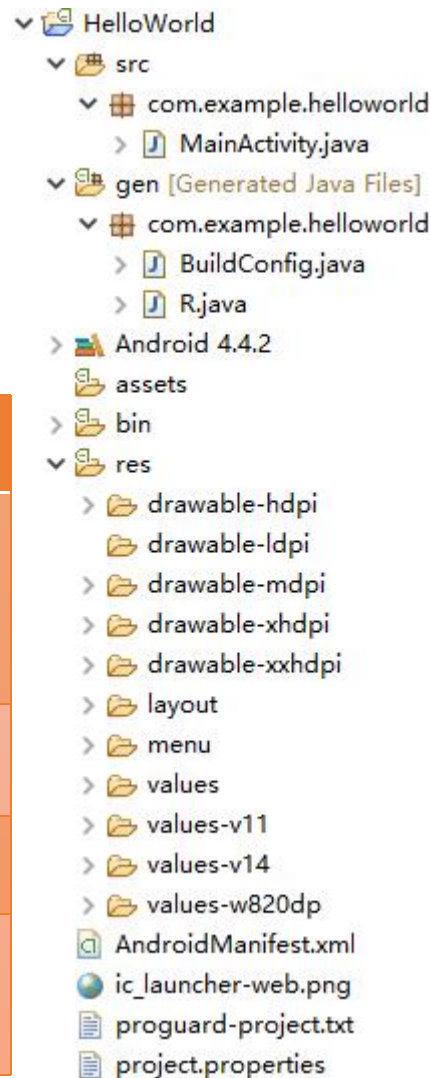


# Android 应用程序工程结构

## ◆ 一个典型的Android App Project，它包含以下几个组成部分

- res文件夹中存放了与App界面相关的资源文件，包括：

子目录	内容说明
drawable-???	分别包含JPEG或者PNG格式的不同尺寸的图形文件，可以使用在具有高、中、低不同分辨率的移动设备屏幕上
layout	包含布局XML文件，例如main.xml
menu	包含定义菜单的XML文件
values	包含定义程序使用的数组、字符串、尺寸、色彩或样式的XML文件，例如strings.xml



# AndroidManifest.xml 文件

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="21" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

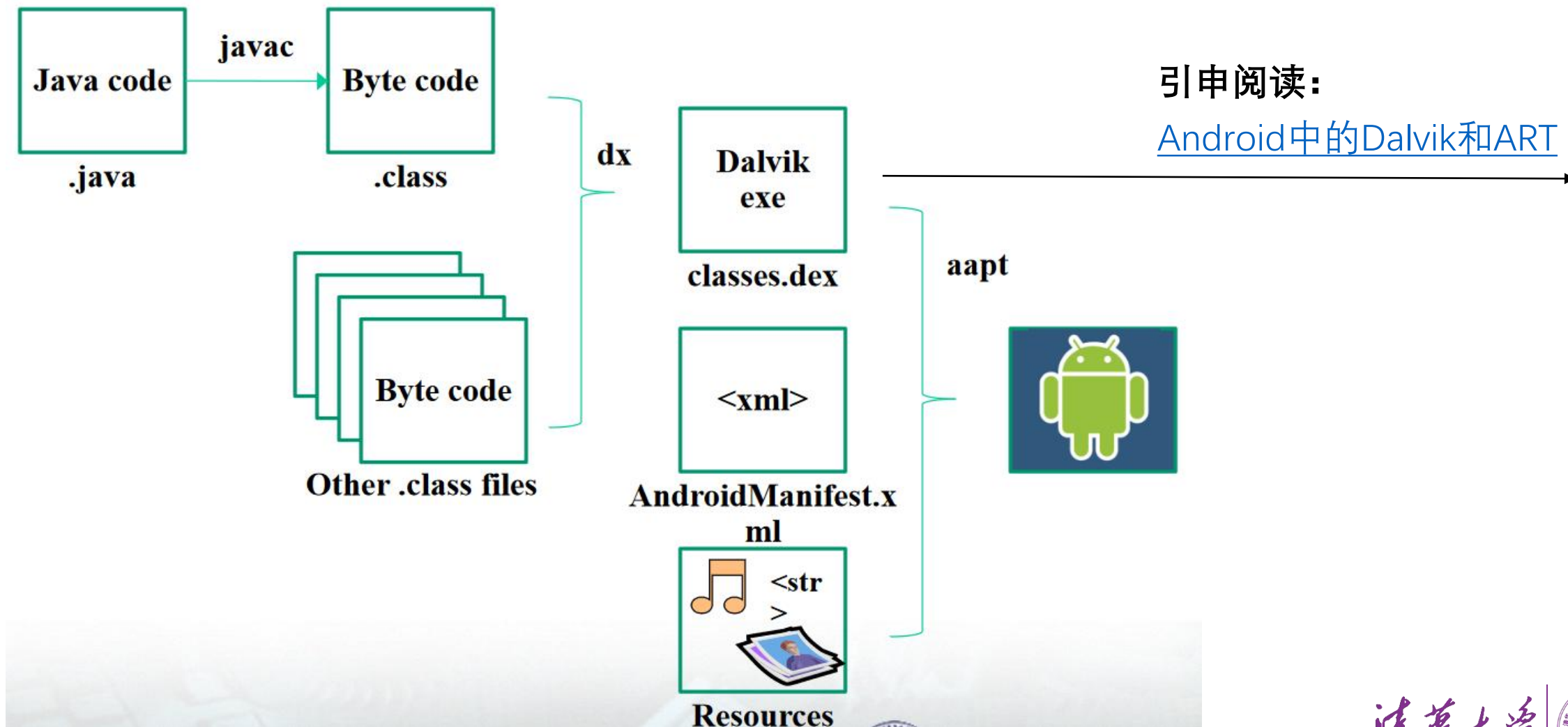
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

- 当Android启动一个应用程序组件之前，它必须知道那个组件是存在的
- 应用程序会在 AndroidManifest.xml 中声明它的组件（包括使用的所有活动、内容提供商、广播接收器和服务控件）
- 这个文件会被打包到Android的应用程序apk中进行发布。

通过<intent-filter>标签，指定程序的主Activity

- 该文件使用XML标记语言进行编写，该语言简单清晰，关于该语言的一些信息，可以参阅：<https://www.w3school.com.cn/xml/index.asp>

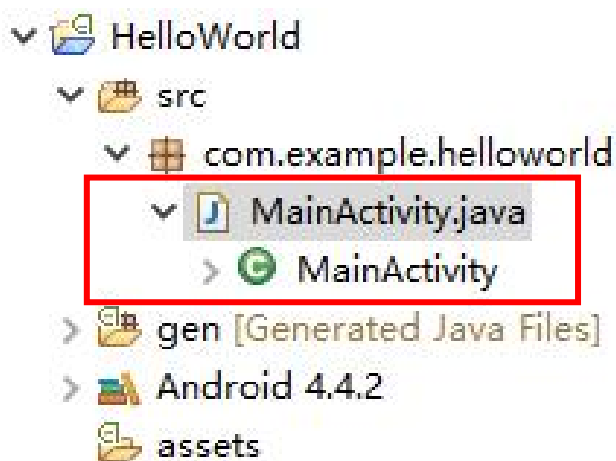
# Android 程序的打包



# Activity 基础

# Activity

- ◆ Activity（活动）是一种可以包含用户界面的组件，主要用于与用户进行交互。
- ◆ 一个应用程序可以包含零个或多个Activity。



```
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.view.Menu;
6 import android.view.MenuItem;
7
8 public class MainActivity extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
```

- ◆ Activity 是 Context 抽象类的子类（非直接子类，而是继承链上的）。我们在写代码的时候，通过继承Activity来实现自己的活动类

# 创建 Activity

- ◆ 使用常规的新建Java Class的方法创建，需要继承android.app.Activity类
- ◆ 重写Oncreate方法

```
public class Activity2 extends Activity {  
    /* (non-Javadoc)  
    * @see android.app.Activity#onCreate(android.os.Bundle)  
    */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // TODO Auto-generated method stub  
    }  
}
```

如何建立Activity 和Layout的绑定关系？



# 注册 Activity

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <manifest android:versionCode="1" android:versionName="1.0"
3     package="com.example.helloworld" xmlns:android="http://schemas.android.com/apk/res/android">
4     <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="21"/>
5     <application android:allowBackup="true"
6         android:icon="@drawable/ic_launcher"
7         android:label="@string/app_name" android:theme="@style/AppTheme">
8         <activity android:label="@string/app_name" android:name=".MainActivity">
9             <intent-filter>
10                 <action android:name="android.intent.action.MAIN"/>
11                 <category android:name="android.intent.category.LAUNCHER"/>
12             </intent-filter>
13         </activity>
14         <!--created by admin at 2019-08-22 15:23:27-->
15         <activity android:name=".Activity2"/>
16     </application>
17 </manifest>
18
```

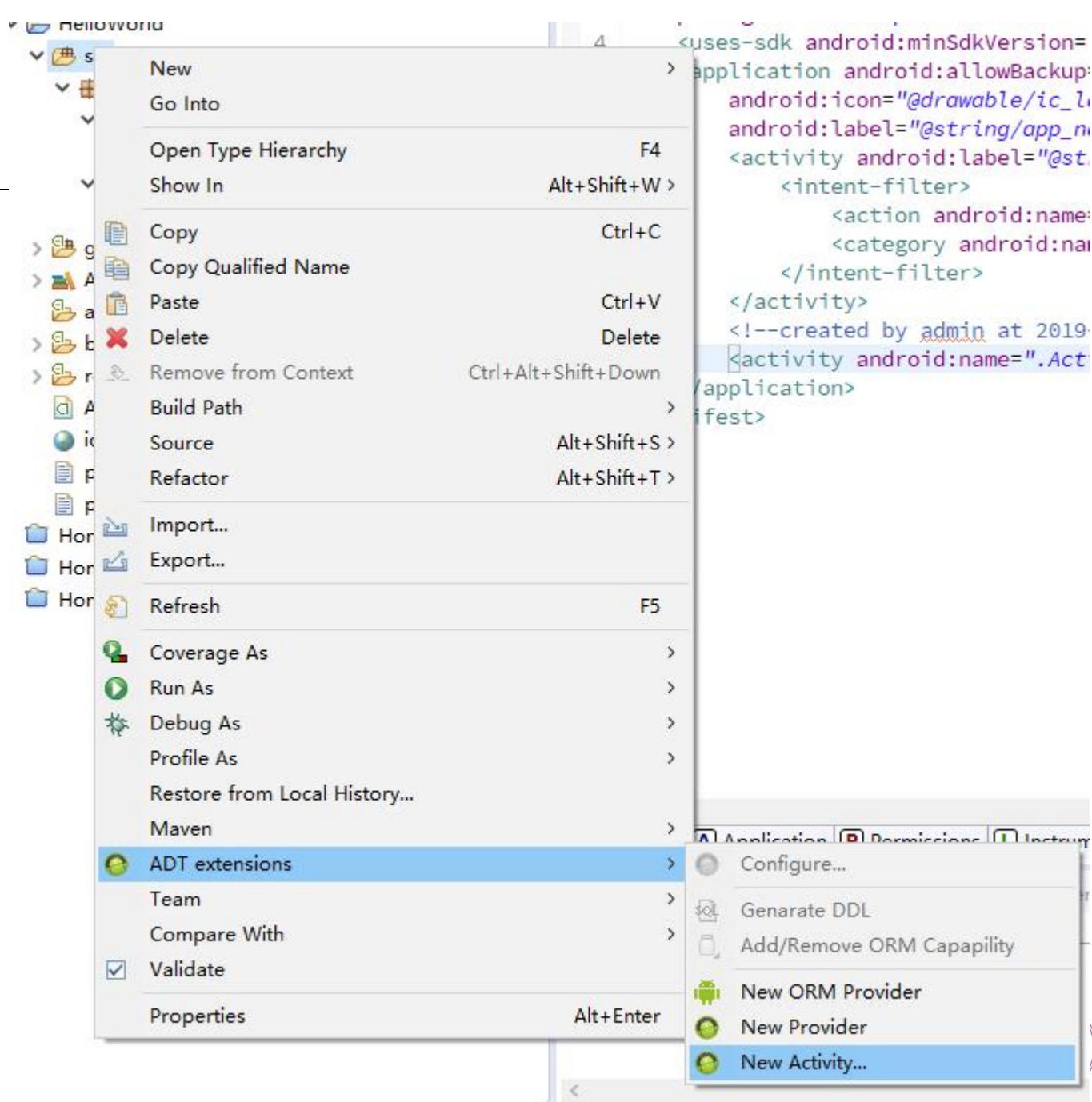


# 创建 Activity方法2

## ◆ 方法（在某些版本可用）

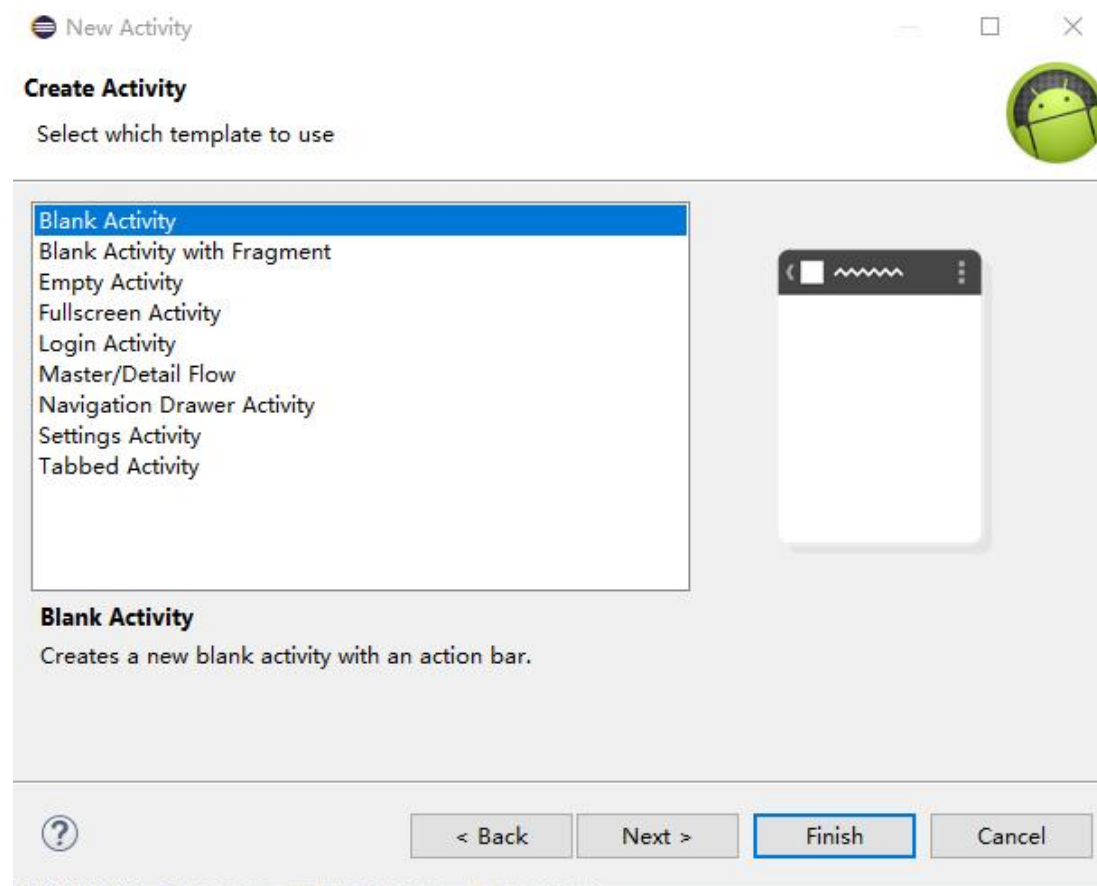
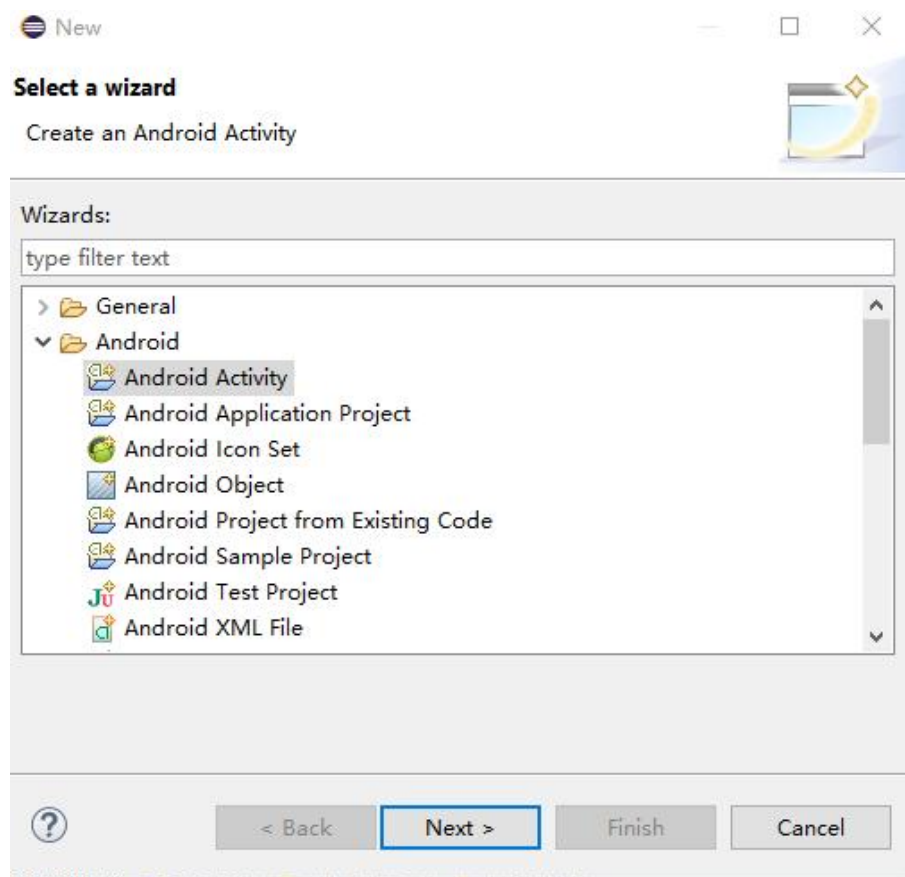
1. 点中src文件夹，右键弹出菜单
2. 进入ADT extensions二级菜单，选择New Activity
3. 在弹出窗口中，设置新Activity的名字即可

这种方法创建的新Activity，会自动在AndroidManifest.xml中注册

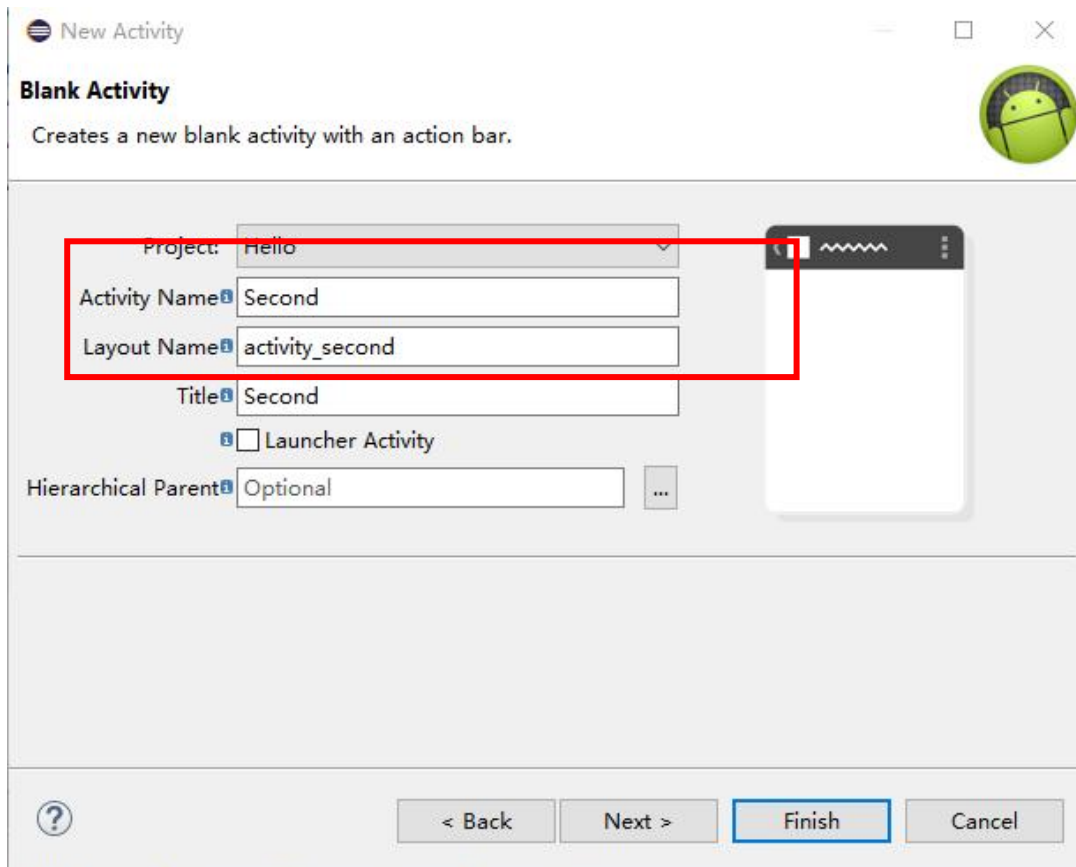


# 创建 Activity方法3

◆ File -> New -> Other, 选择 Android Activity



# 创建 Activity方法3



```
public class Activity2 extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_second);  
    }  
}
```

```
17         android:name=".MainActivity"  
18         android:label="@string/app_name" >  
19         <intent-filter>  
20             <action android:name="android.intent.action.MAIN" />  
21             <category android:name="android.intent.category.LAUNCHER" />  
22         </intent-filter>  
23     </activity>  
24     <activity  
25         android:name=".Second"  
26         android:label="@string/title_activity_second" >  
27     </activity>  
28 </application>  
29 </manifest>  
30  
31 </manifest>  
32
```

# 练一练

---

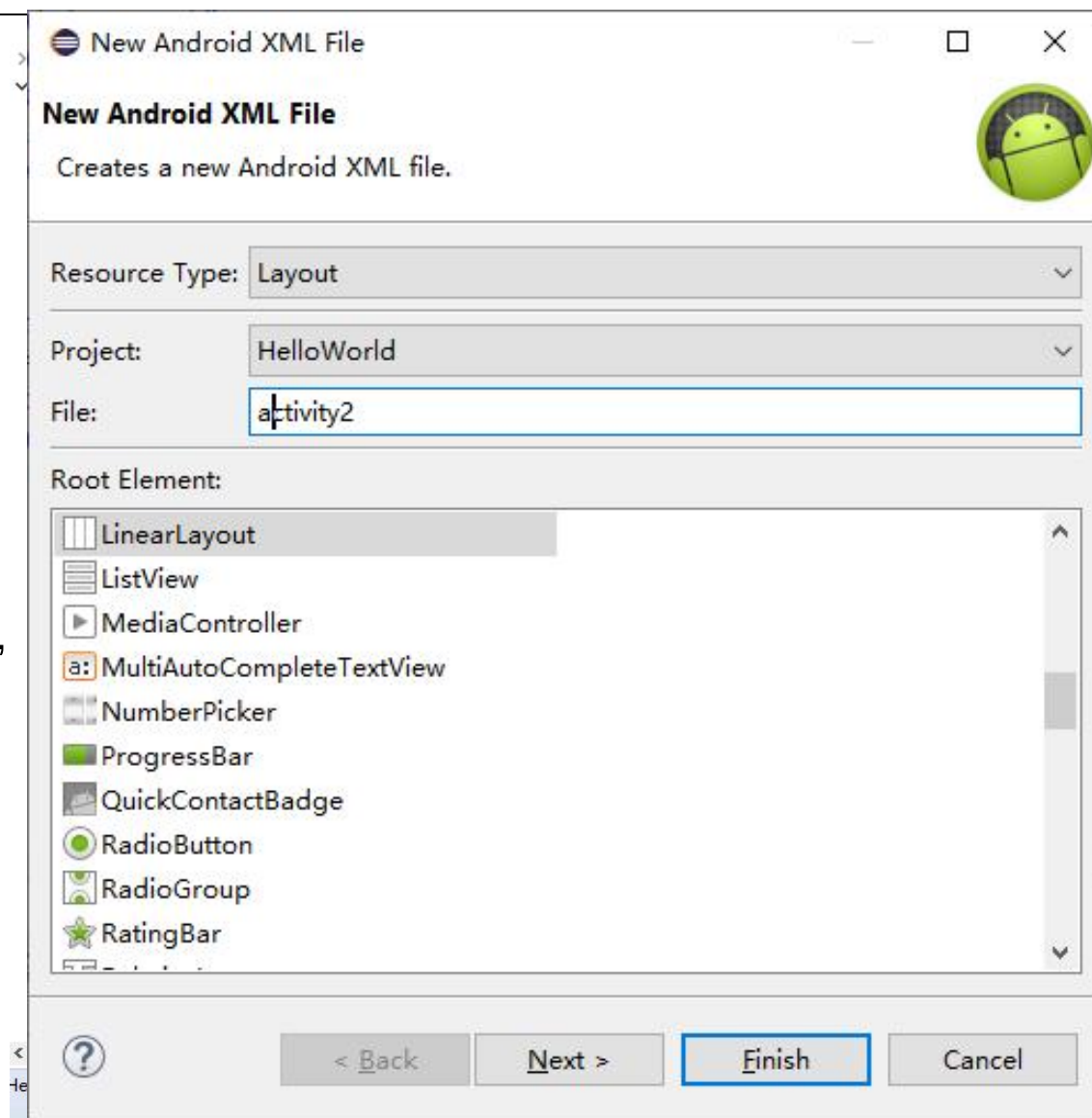
采用方法2/3实现

1. 在HelloWorld中新建新的活动Activity3
2. 将新添加的Layout加载到Activity3中

# 添加布局(Layout)到活动 (Activity)

- ◆ 布局 (Layout) 对于Android程序设计来说是一件十分重要的工作，布局定义了我们的界面的外观和放置的所有控件
- ◆ 在Eclipse中，我们可以使用ADT插件快速创建布局文件 (XML)
  - **右键项目名称**，进入Android Tools二级菜单，选择New Resource File
  - 或者File -> New->Other  
选择 Android XML Layout file

◆ **注意Layout文件名首字母必须小写**



# 添加布局(Layout)到活动 (Activity)

- ◆ 在添加布局文件之后，我们可以将该布局文件添加到刚刚新建的Activity中去

```
13 public class Activity2 extends Activity {  
14  
15     /* (non-Javadoc)  
16      * @see android.app.Activity#onCreate(android.os.Bundle)  
17      */  
18     @Override  
19     public void onCreate(Bundle savedInstanceState) {  
20         super.onCreate(savedInstanceState);  
21         setContentView(R.layout.activity2);  
22     }  
23 }  
24  
25 }  
26
```

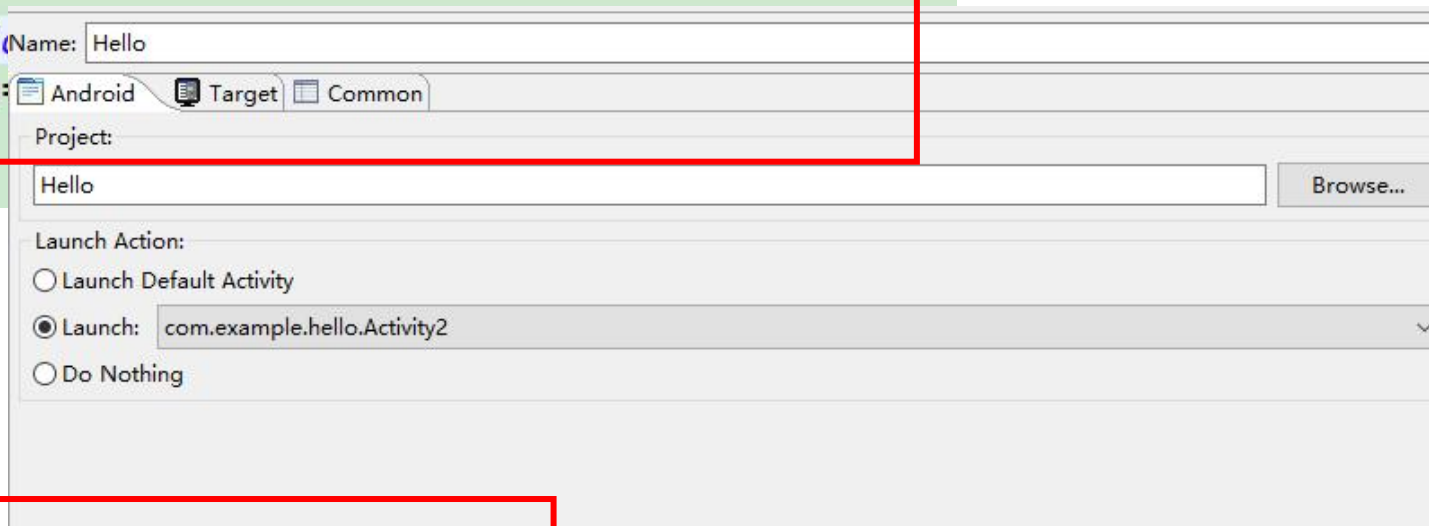
建立Layout和Activity的关联关系

这样我们就可以使用res\layout\activity2.xml来设置Activity2这个活动的界面了。



# 修改启动配置

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name=".Activity2" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```



(Name: Hello

Android Target Common

Project:

Hello Browse...

Launch Action:

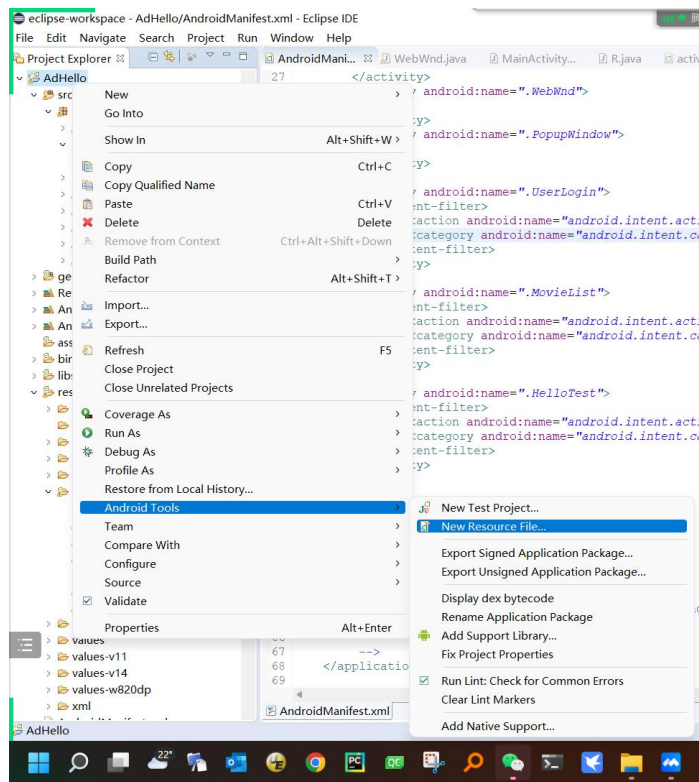
☐ Launch Default Activity

☒ Launch: com.example.hello.Activity2

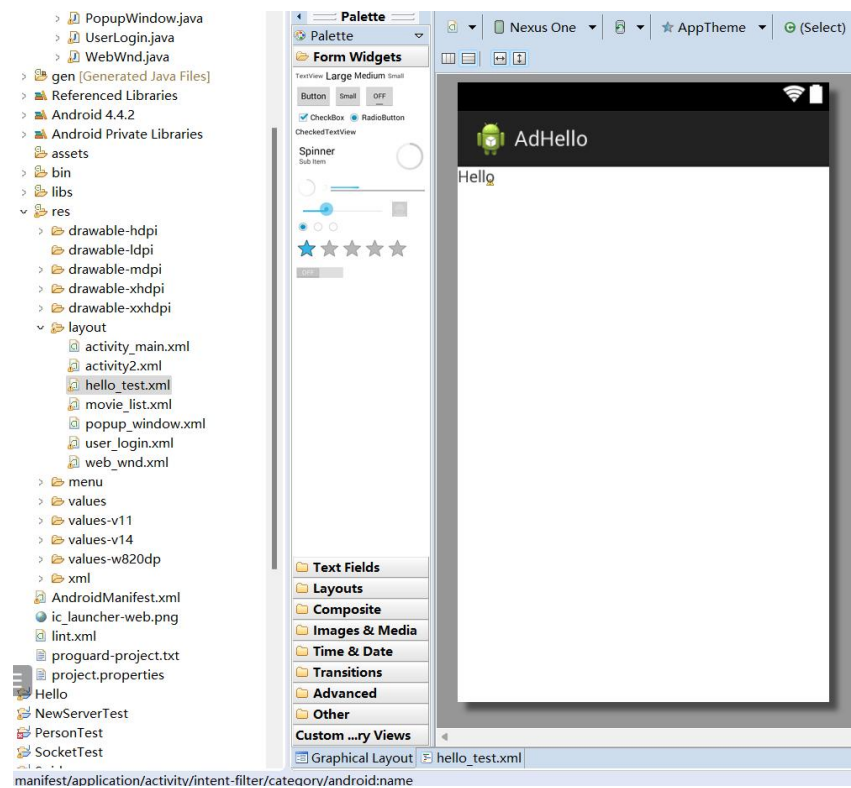
☐ Do Nothing

# 课堂练习(1)

◆ 新建一个activity界面，并可以作为主界面启动。注意，需要修改的内容：



新建页面

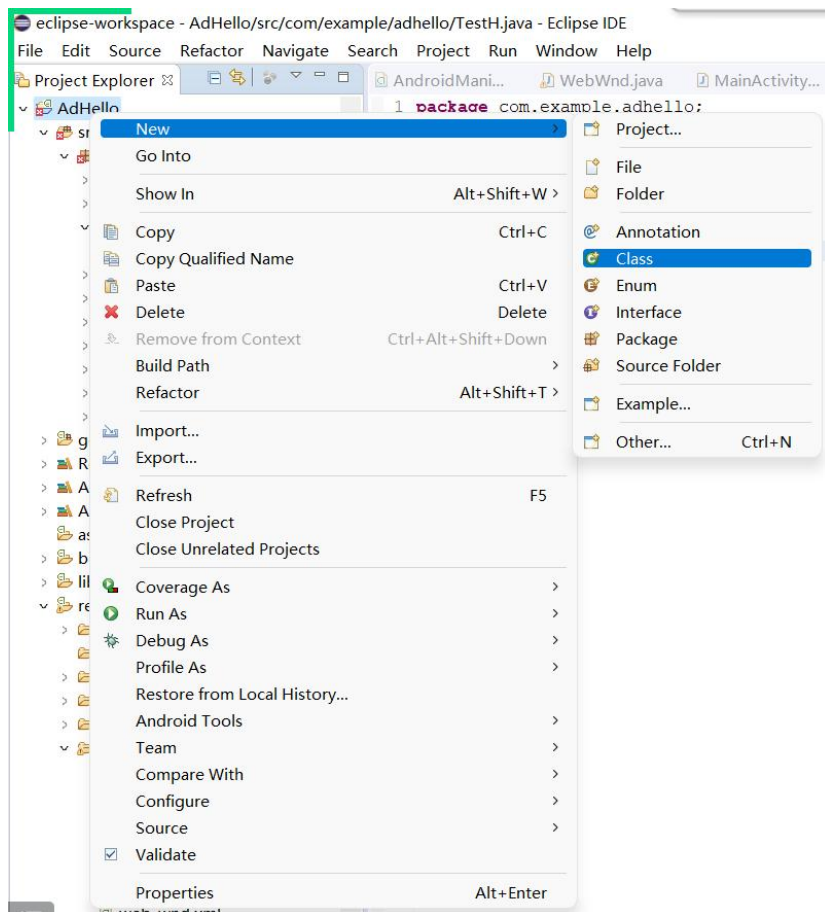


修改页面

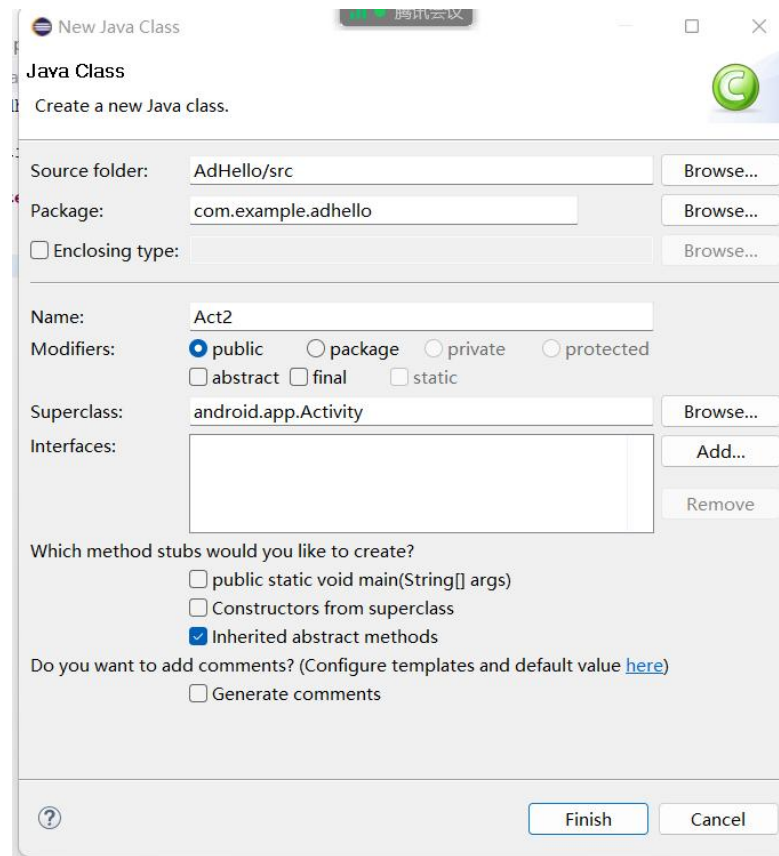


# 课堂练习(1)

- ◆ 新建一个activity界面，并可以作为主界面启动。注意，需要修改的内容：



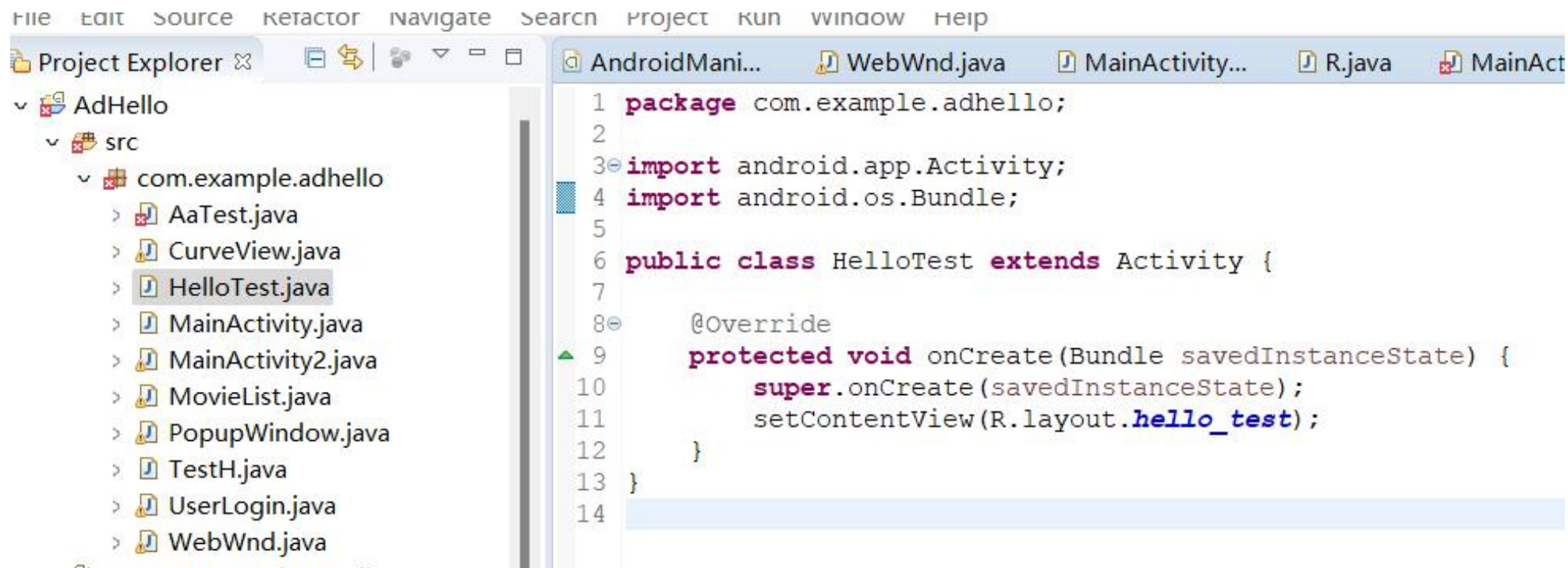
新建类



设置类名称与父类

# 课堂练习(1)

◆ 新建一个activity界面，并可以做为界面启动。注意，需要修改的内容：



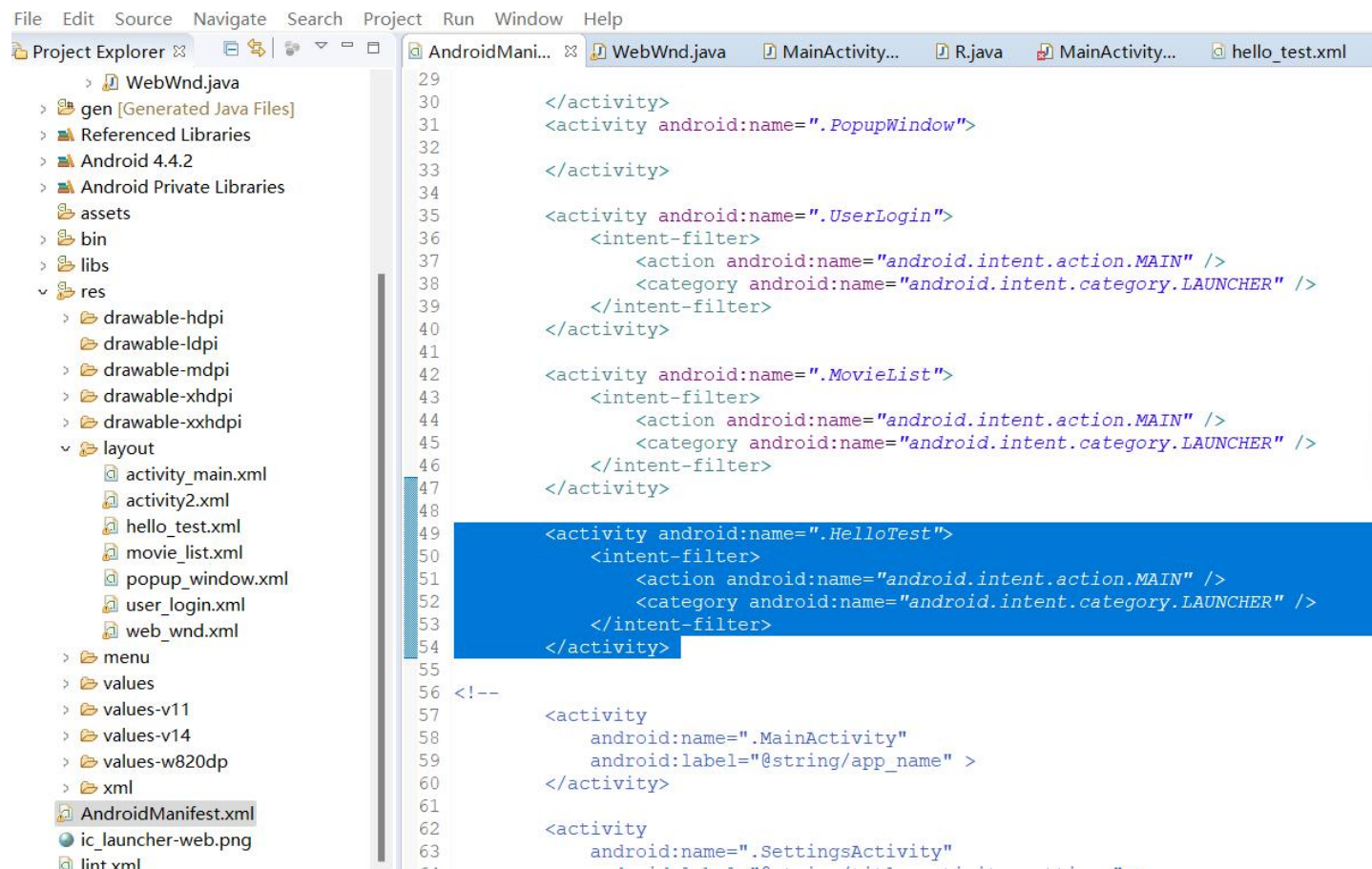
The screenshot shows the Android Studio interface. On the left, the 'Project Explorer' pane displays a project named 'AdHello' with a source folder 'src' containing a package 'com.example.adhello'. Several Java files are listed, with 'HelloTest.java' selected. The main editor on the right shows the code for 'HelloTest.java'. The code is as follows:

```
1 package com.example.adhello;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5
6 public class HelloTest extends Activity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.hello_test);
12     }
13 }
14
```

修改类

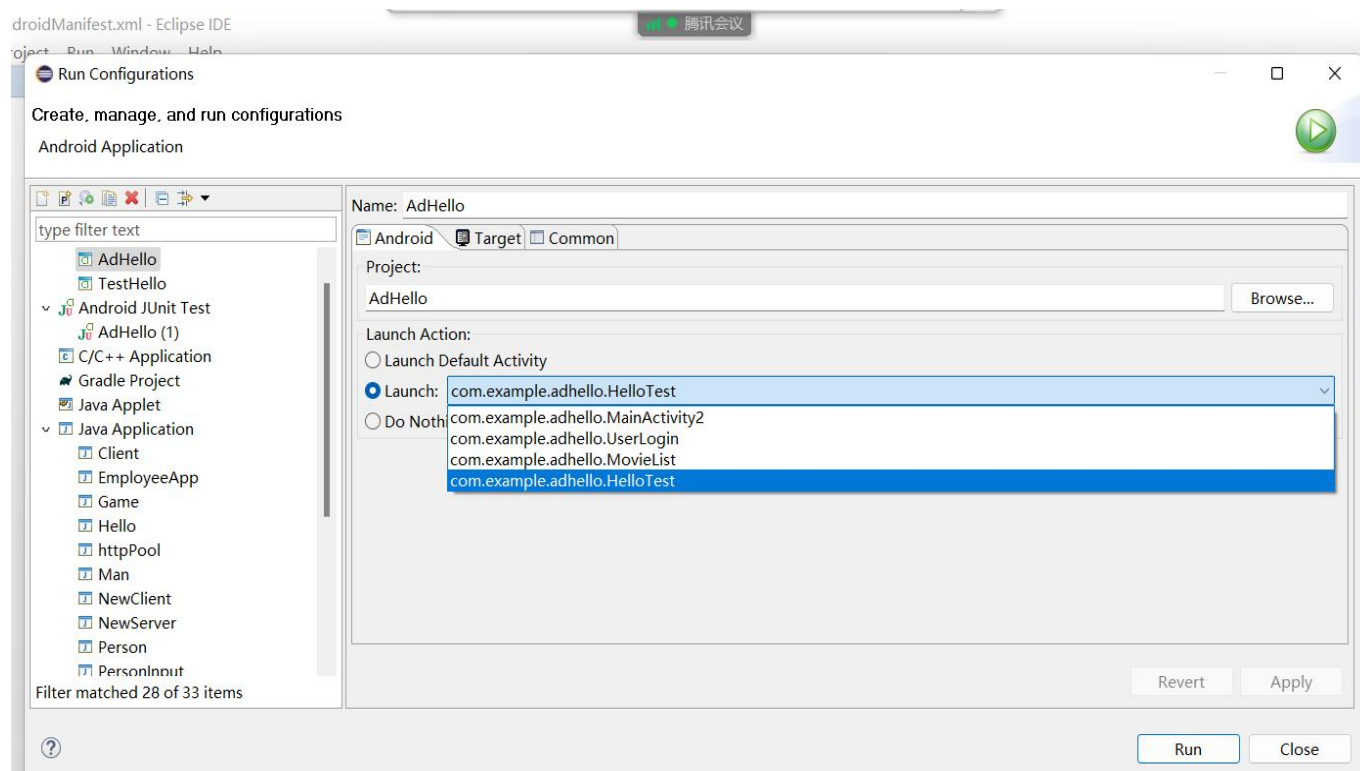
# 课堂练习(1)

- ◆ 新建一个activity界面，并可以作为主界面启动。注意，需要修改的内容：



修改AndroidManifest.xml，添加activity

◆ 新建一个activity界面，并可以做为界面启动。注意，需要修改的内容：



清华大学  
Tsinghua University

# 常用布局



# 常用布局 Layout

---

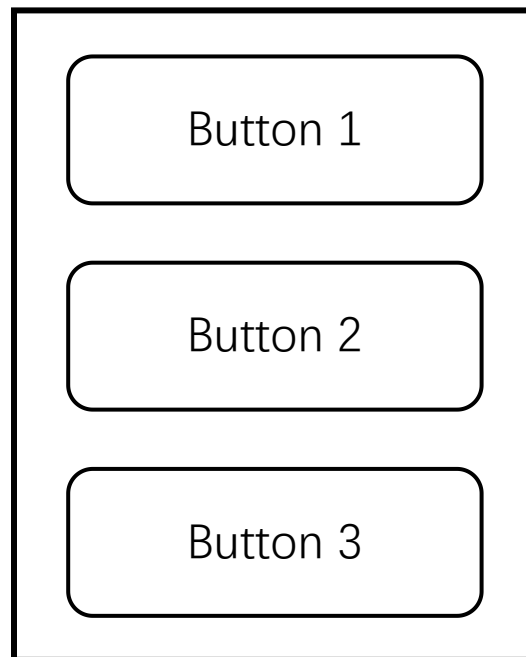
- ◆ 在前面一章中我们讲了如何新建一个Layout并将其绑定到一个Activity上  
那么常用的Layout有哪些？
- ◆ Android提供多种布局对象，用来编排子界面上的组件（Views），常见的几种布局类如下所示（ViewGroup是所有布局的基类）
  - LinearLayout类：使用该类布局的子界面控件是一个接着一个排列成水平或垂直的一条直线，在xml中使用<LinearLayout>标签来使用。
  - RelativeLayout类：允许子界面控件通过指定相对与其他子界面控件的相对位置来确定自己的位置。在xml中使用<RelativeLayout>标签来使用。
  - FrameLayout类：如同堆栈来编排多个子界面控件，所有子控件的起始位置都是左上角的同一个位置，通过标签来进行选择，在xml中使用<FrameLayout>标签来使用。
  - TableLayout类：使用表格来编排子界面控件，每一个界面控件都有对应的行与列，在xml中使用<TableLayout>标签和<TableRow>标签来使用。

# LinearLayout布局

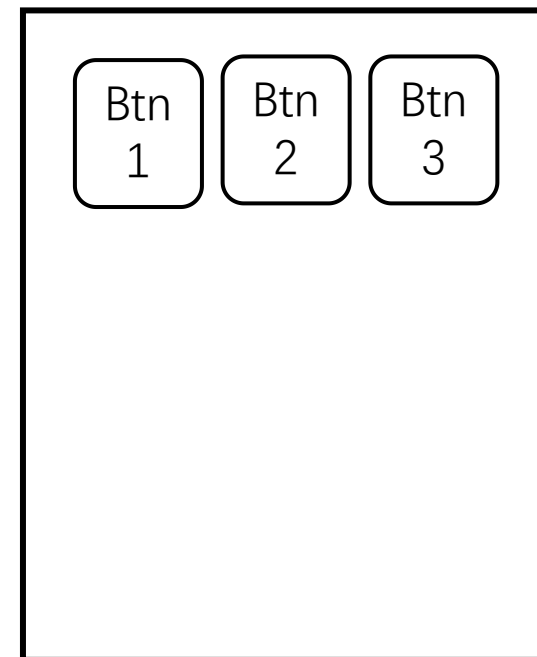
- ◆ LinearLayout 布局是最常使用的布局方式，它可以将子界面控件排成一行（垂直），或一行（水平），一个接着一个排列成直线，取其线性的含义。

- ◆ XML 代码

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    .....
</LinearLayout>
```



垂直排布



水平排布

# LinearLayout布局

## ◆ 常用属性

LinearLayout的常用属性及其说明如下

属性	说明
Orientation	指定布局的方向，vertical（垂直）或horizontal（水平）
layout_width	设定容器的宽度
layout_height	设定容器的高度
layout_weight	在包含的子控件上加上此属性，可以指定控件的重要性，例如，3个Button分别是0.25、0.5和0.25，表示中间控件是其他的两倍大

在layout\_width和layout\_height属性中，可以使用fill\_parent、match\_parent和wrap\_content来设置，也可以使用**数字+单位**（表示大小）来设置



# LinearLayout布局

## ◆ fill\_parent、match\_parent 和 wrap\_content

- wrap\_content : wrap 翻译过来是包裹，content是内容。那么这个就是包裹内容的意思，也就是说你的控件里面的内容有多大，这个控件就有多大。
- fill\_parent 和 match\_parent :让作用的控件**填满父容器的其他空间**。有点类似C#布局中的Dock属性。

## ◆ Android 使用的尺寸单位包括

单位	说明
dp或者dip	Desity-independent Pixel的简称，这是Android建议使用的单位
sp	Scale-independent Pixel，类似dp，建议在字型尺寸中使用
pt	Point，点，一点等于1/72英寸
px	实际屏幕上的像素点，Android不建议使用该尺寸单位

# LinearLayout布局

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="老子是Button1"
    />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="老子是Button2"
    />
</LinearLayout>
```



- ◆ 使用线性布局LinearLayout垂直摆放两个按钮，如果上面按钮的高度是fill\_parent或者match\_parent，那么下面的按钮根本显示不出来。

# LinearLayout布局

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical" >
6     <Button
7         android:layout_width="fill_parent"
8         android:layout_height="wrap_content"
9         android:text="老子是Button1"
10    />
11     <Button
12         android:layout_width="fill_parent"
13         android:layout_height="fill_parent"
14         android:text="老子是Button2"
15    />
16
17 </LinearLayout>
```



- ◆ 反之，上面是wrap\_content，下面是fill\_parent或者match\_parent，那么上面会有，下面的这个按钮会填充剩余的其他部分。

# LinearLayout布局

## 支持多层嵌套

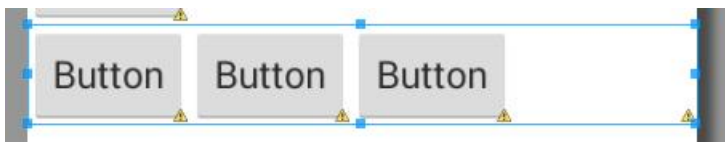
```
<LinearLayout android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <Button android:text="按钮四"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.25"/>
    <Button android:text="按钮五"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.75"/>
</LinearLayout>
```



- 按比例划分，对屏幕适配
- 太多层嵌套会占用更多的系统资源,还有可能引发stackoverflow

# LinearLayout布局-横向的左-中-右

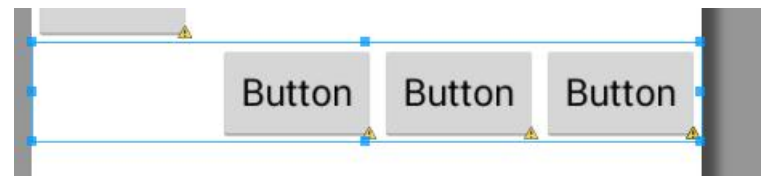
```
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:gravity="left"  
android:orientation="horizontal"
```



```
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:gravity="center"  
android:orientation="horizontal"
```

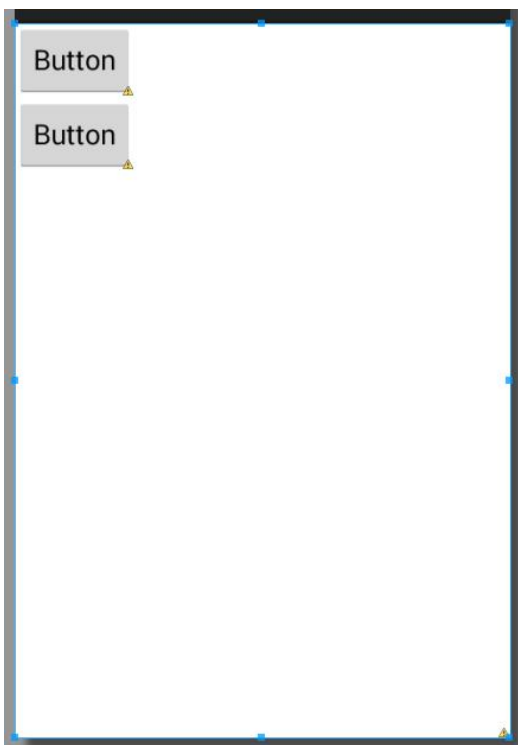


```
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:gravity="right"  
android:orientation="horizontal"
```

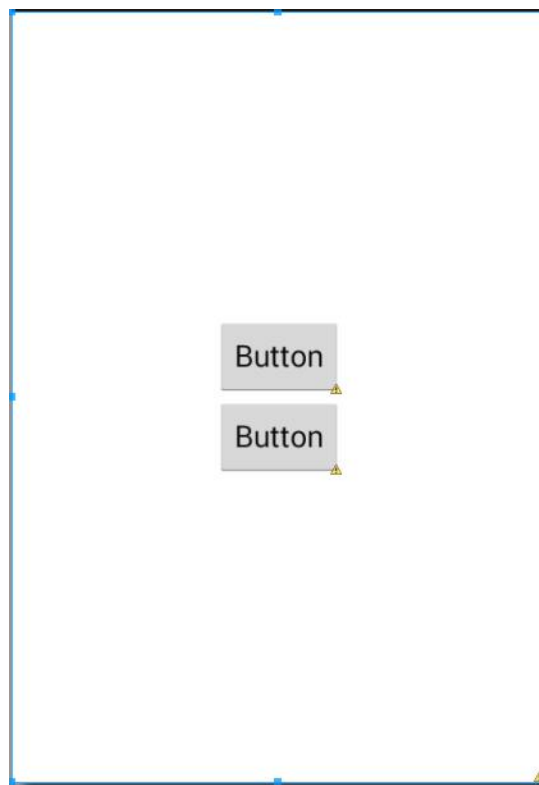


# LinearLayout布局-纵向的上-中-下

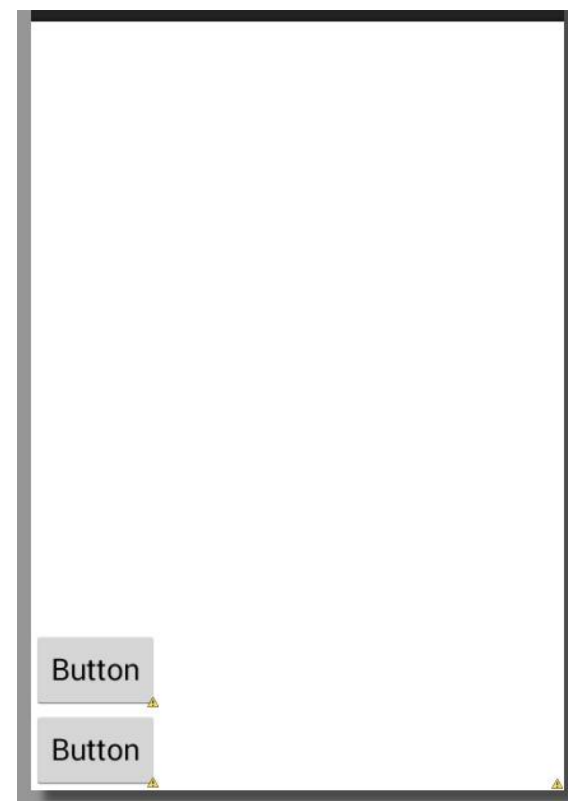
```
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:orientation="vertical"  
android:gravity="top"
```



```
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:orientation="vertical"  
android:gravity="center"
```



```
android:layout_width="match_parent"  
android:layout_height="match_parent"  
android:orientation="vertical"  
android:gravity="bottom"
```



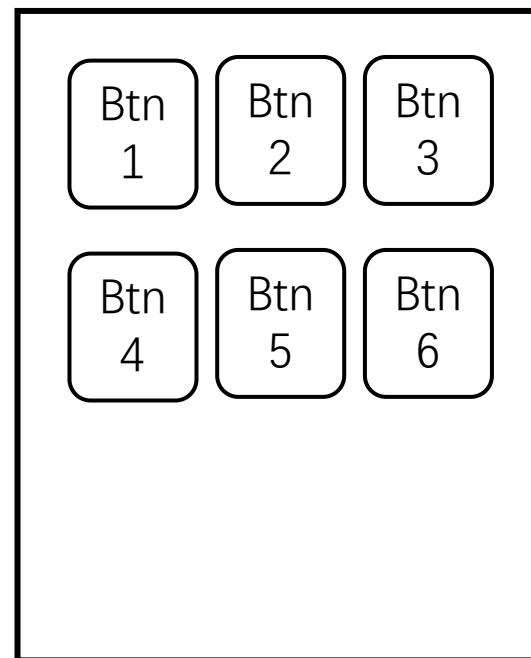
# TableLayout布局

## ◆ TableLayout 布局使用表格方式来编排子界面控件

在 TableLayout 中，使用 TableRow 元素来定义每一行的界面控件

## ◆ XML 代码

```
<TableLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TableRow ...>
        .....
    </TableRow>
</TableLayout>
```



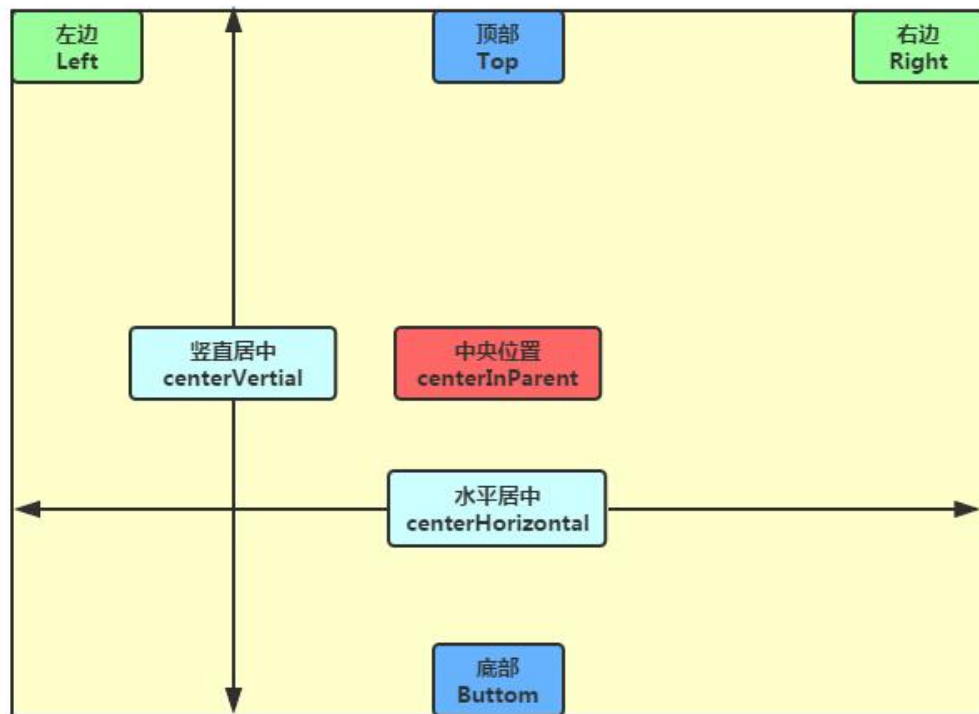
# TableLayout布局

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TableRow>
        <Button android:text="按钮1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <Button android:text="按钮2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </TableRow>
    <TableRow>
        <Button android:text="按钮3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <Button android:text="按钮4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </TableRow>
</TableLayout>
```





# RelativeLayout (相对布局)

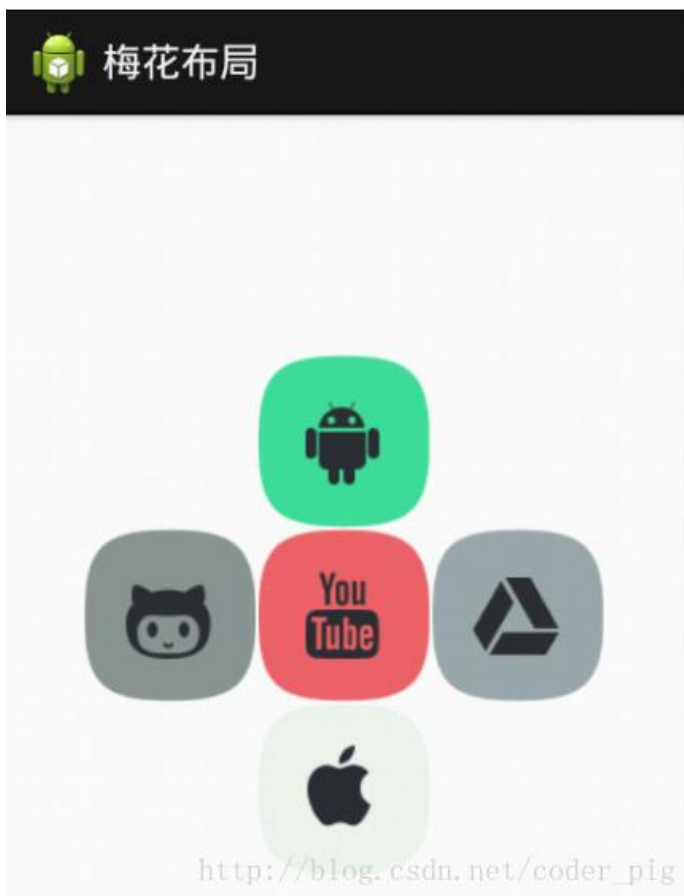


根据父容器定位 [g. csdn. net/coder\\_pig](http://blog.csdn.net/coder_pig)



根据兄弟组件定位

# RelativeLayout (相对布局)



<!-- 这个是在容器中央的 -->

```
<ImageView
    android:id="@+id/img1"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_centerInParent="true"
    android:src="@drawable/pic1"/>
```

<!-- 在中间图片的左边 -->

```
<ImageView
    android:id="@+id/img2"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_toLeftOf="@id/img1"
    android:layout_centerVertical="true"
    android:src="@drawable/pic2"/>
```

<!-- 在中间图片的右边 -->

```
<ImageView
    android:id="@+id/img3"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_toRightOf="@id/img1"
    android:layout_centerVertical="true"
    android:src="@drawable/pic3"/>
```

<!-- 在中间图片的上面 -->

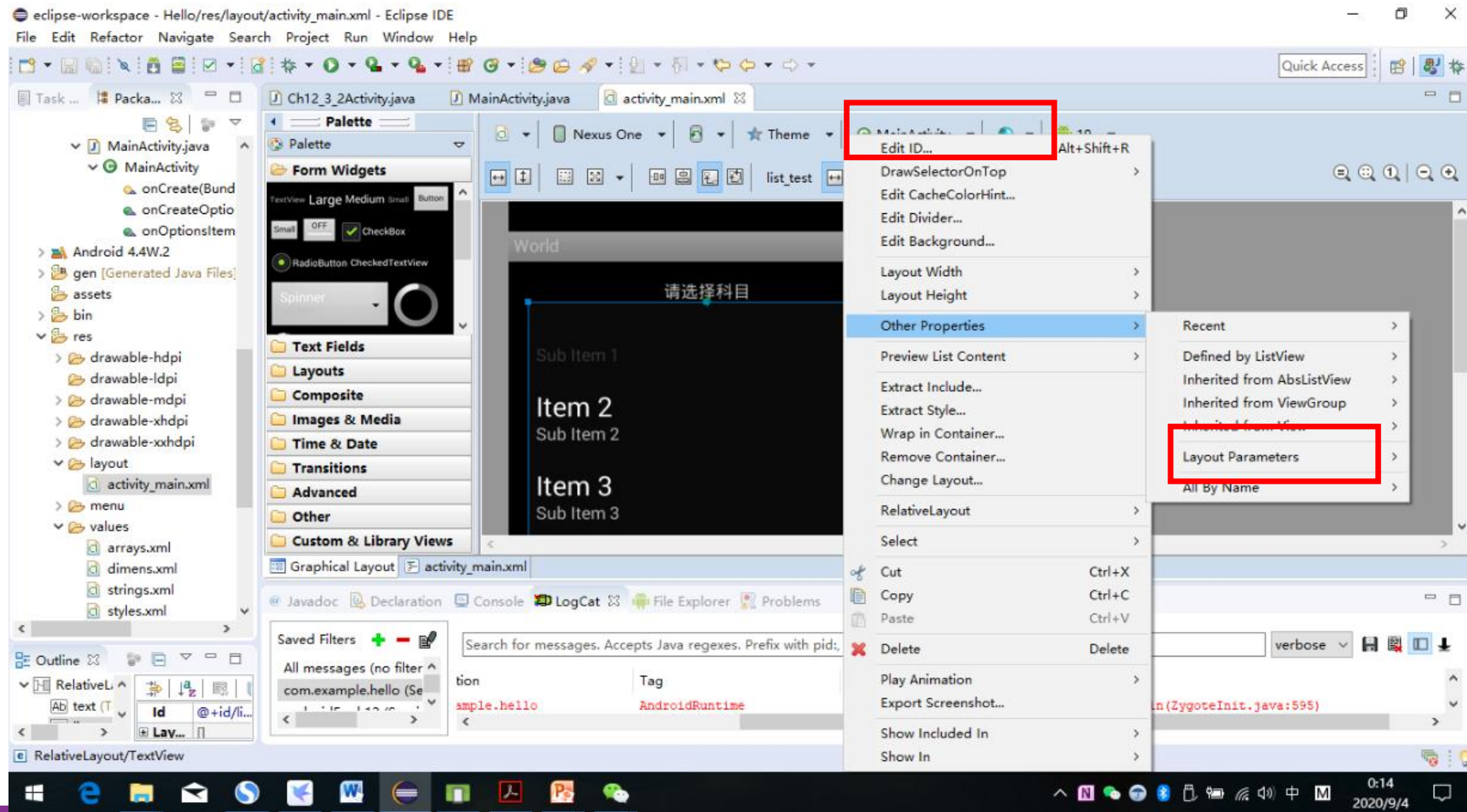
```
<ImageView
    android:id="@+id/img4"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_above="@id/img1"
    android:layout_centerHorizontal="true"
    android:src="@drawable/pic4"/>
```

<!-- 在中间图片的下面 -->

```
<ImageView
    android:id="@+id/img5"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_below="@id/img1"
    android:layout_centerHorizontal="true"
    android:src="@drawable/pic5"/>
```

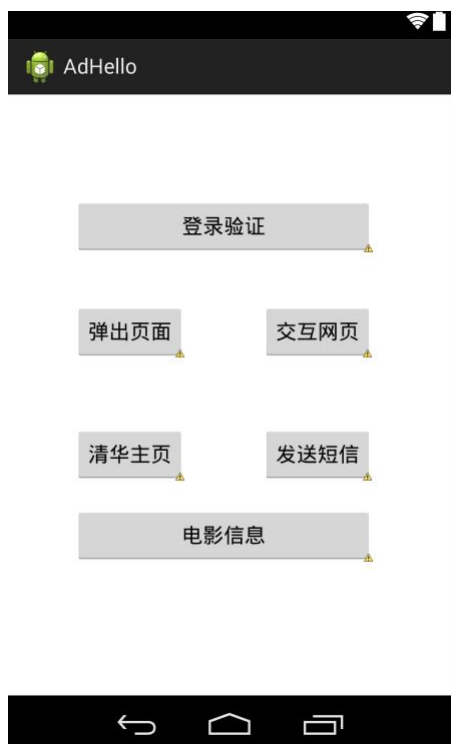
练一练

# Eclipse修改layout文件

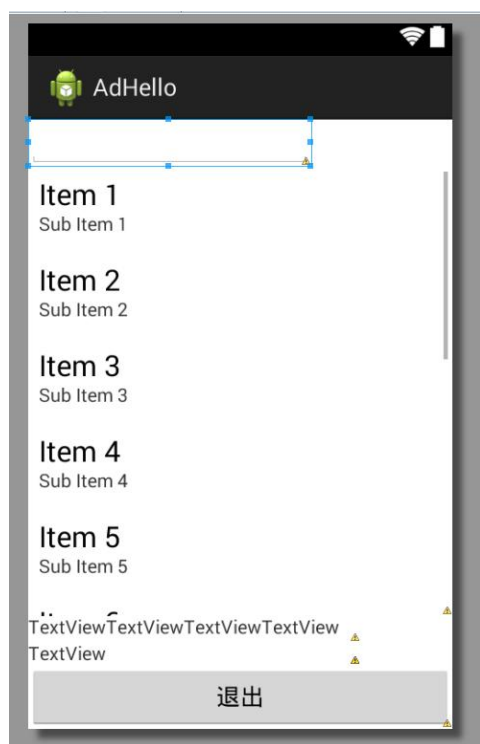


# 课堂练习(2)

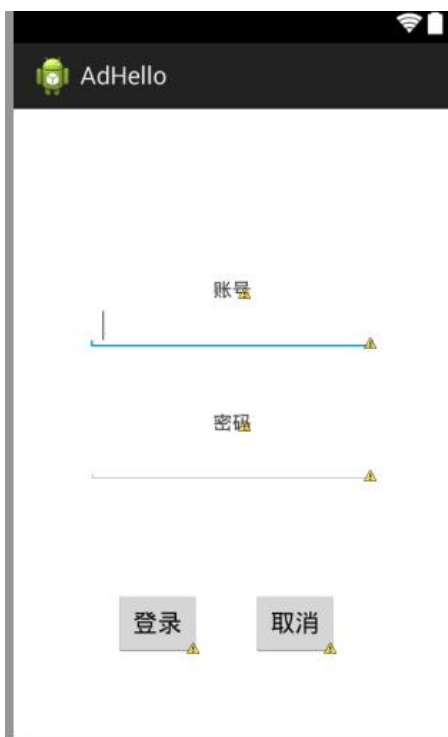
◆ 依次实现如下布局：



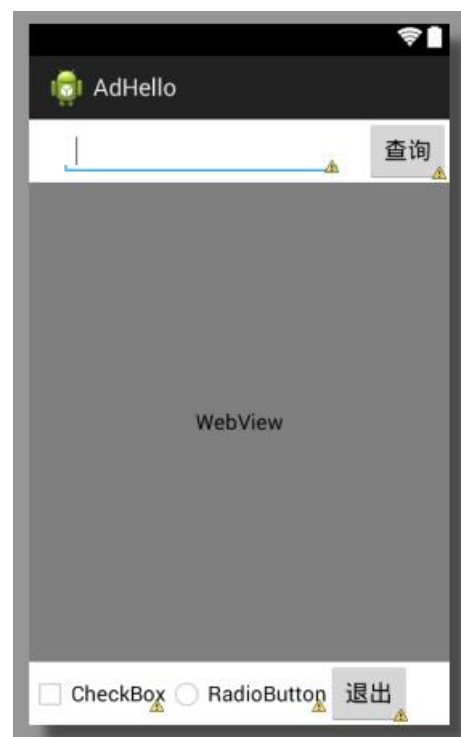
ActivityMain



MovieList



UserLogin

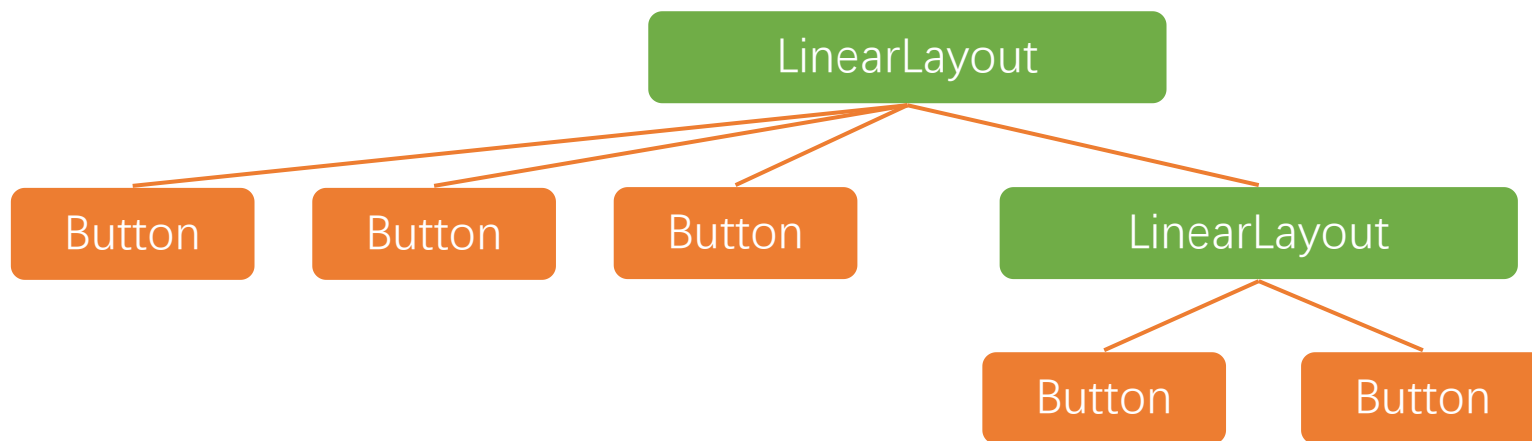


WebPage

# Android 控件基础

# View 与 ViewGroup 类

- ◆ View 类是所有用户界面控件的基础类（直接或者间接父亲），其继承子类分为两大类：
  - 界面控件（Widgets，可称为View对象）：正确地说，Android的界面控件都称为Widget，例如，Button和EditText控件等等
  - 布局类（Layout Class，可称为View Group对象）：ViewGroup 抽象类是View的子类，它是布局类的父亲，可以用来组织其他界面控件
- ◆ 界面组织可以使用嵌套的方法，比如





# TextView 控件

- ◆ TextView 控件就是标签控件，它是用于数据输出控件，可以显示消息正文或者程序的执行结果，在布局资源的XML中使用<TextView>来声明。如下所示：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="摄氏"
>
```

- ◆ 其中 android:text 属性就指明了该控件显示的内容，在android开发中，字符串更推荐定义在res/values/string.xml内，然后在空间属性中引用字符串的名字来使用它



# EditText 控件

- ◆ EditText 是 TextView 的子类，用于输入程序所用的数据

```
<EditText
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:inputType="number"
>
```

- ◆ 其中 android:inputType 属性指明该控件输入的类型，4种常用类型包括：

类型	说明	类型	说明
none	只读，不能输入	number	整数
text	一般文字	time	时间

等等，可选类型非常多，同学们可以自行查阅资料

# Button 控件

---

- ◆ 顾名思义，Button 控件就是一个按钮，可以触发Click事件（事件处理方法在后面我们会进一步介绍）
- ◆ Button常用的两个属性如下表所示

属性	说明
android:text	按钮上的文字
android:onClick	指定Button控件的事件处理方法

# ListView 控件



```
<TextView
    android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:text="@string/hello_world"/>
```

```
<ListView
    android:id="@+id/list_test"
    android:layout_below="@id/text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</ListView>
```

# ListView 控件

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

    //要显示的数据
    String[] strs = {"语文", "数学", "英语", "物理", "化学"};
    //创建ArrayAdapter
    ArrayAdapter<String> adapter = new ArrayAdapter<String>
        (this, android.R.layout.simple_expandable_list_item_1, strs);
    //获取ListView对象, 通过调用setAdapter方法为ListView设置Adapter设置适配器
    ListView list_test = (ListView) findViewById(R.id.list_test);
    list_test.setAdapter(adapter);
}
```

# 作业





# 提示:

---

## 背景颜色

```
<RelativeLayout
    android:id="@+id/title"
    android:layout_width="fill_parent"
    android:layout_height="45.0dip"
    android:background="#ffcd2626"
    android:gravity="center_vertical">
```

## 背景图片

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_below="@id/title"
    android:background="@drawable/biz_plugin_weather_shenzhen_bg"
    android:orientation="vertical" >
```

## 推荐资源

<https://www.runoob.com/w3cnote/android-tutorial-relativelayout.html>