

android  
developers

# Android开发教程&笔记



android

## Android 应用开发 3

### 使用 Bundle 在 Activity 间传递数据

#### 从源 Activity 中传递数据



//数据写入 Intent

```
Intent openWelcomeActivityIntent=new Intent();
Bundle myBundelForName=new Bundle();
myBundelForName.putString("Key_Name",inName.getText().toString());
myBundelForName.putString("Key_Age",inAge.getText().toString());
openWelcomeActivityIntent.putExtras(myBundelForName);
openWelcomeActivityIntent.setClass(AndroidBundel.this, Welcome.class);
startActivity(openWelcomeActivityIntent);
```

#### 目标 Activity 中获取数据

//从 Intent 中获取数据

```
Bundle myBundelForGetName=this getIntent().getExtras();
String name=myBundelForGetName.getString("Key_Name");
myTextView_showName.setText("欢迎您进入: "+name);
```


### 使用 Bundle 在 Activity 间传递数据 2

#### 从源请求 Activity 中通过一个 Intent 把一个服务请求传到目标 Activity 中

```
private Intent toNextIntent;//Intent 成员声明
toNextIntent=new Intent();//Intent 定义
toNextIntent.setClass(TwoActivityME3.this, SecondActivity3.class);
//设定开启的下一个Activity
startActivityForResult(toNextIntent, REQUEST_ASK);
//开启 Intent 时候，把请求码同时传递
```



在源请求 Activity 中等待 Intent 返回应答结果，通过重载 onActivityResult()方法



```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
    // TODO Auto-generated method stub
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode==REQUEST_ASK){
        if(resultCode==RESULT_CANCELED){
            setTitle("Cancel****");
        }else if(resultCode==RESULT_OK){
            showBundle=data.getExtras();//从返回的Intent中获得Bundle
            Name=showBundle.getString("myName");//从bundle中获得相应数据
            text.setText("the name get from the second layout:\n"+Name);
        }
    }
}
```

- ☺ 第一个参数是你开启请求Intent时的对应请求码，可以自己定义。
- ☺ 第二个参数是目标Activity返回的验证结果码
- ☺ 第三个参数是目标Activity返回的Intent

目标 Activity 中发送请求结果代码，连同源 Activity 请求的数据一同绑定到 Bundle 中通过 Intent 传回源请求 Activity 中

```
backIntent=new Intent();
stringBundle=new Bundle();
stringBundle.putString("myName", Name);
backIntent.putExtras(stringBundle);
setResult(RESULT_OK, backIntent);//返回Activity结果码
finish();
```

## Log 与 DDMS(查看 Log 等信息)

```
Log.v("TAG", "nextPage_Activity onStart()");//设置标签来跟踪程序
```



## Activity 生命周期

### Activity 状态

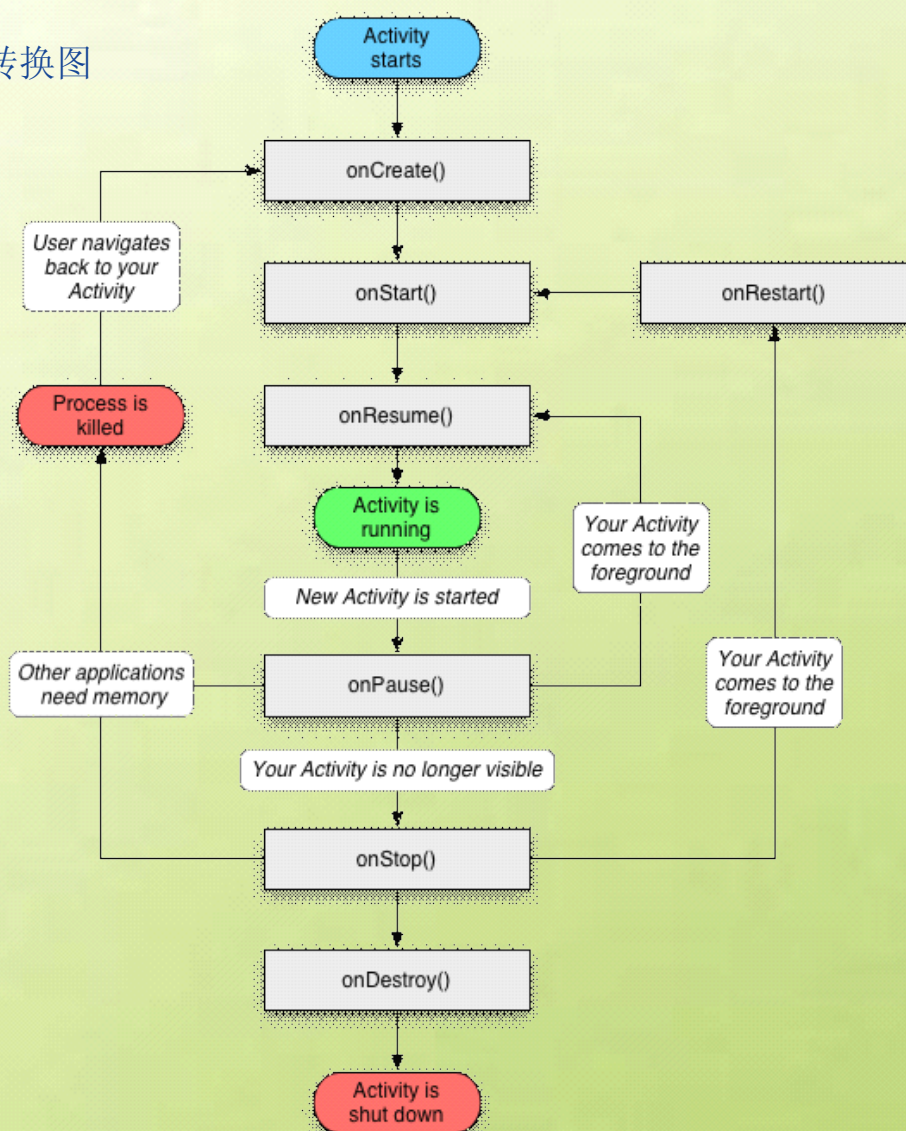
① 当一个 Activity 在屏幕的最上层时（对堆栈的最顶端），它就是属于 active 或者 running 的状态

② 如果一个 Activity 失去焦点（focus）但还看得到它的画面（比如：一个新的 Activity 画面并不是全屏或者它是一个半透明的情况），那失去焦点的 Activity 则处在 paused 的状态。像这个失去焦点的 Activity 它还是完全活着的，并没有消失。（活着的意思是指，Activity 自己本身所有的状态及数据都还是存在的，也跟窗口管理程序 window manager 保持联系着），像这种 paused 的 Activity，会在一种情况下消失，那就是当系统的内存不够用之时，系统会自动判断，八部重要的 Activity 移除。

③ 如果一个 Activity 被其它的 Activity 完全的遮盖住时，它仍然保有全部的状态及数据，但因为它已不再被使用者看见，所以它的画面是被隐藏起来的（画面不需要更新），当系统内存不足时，这种 stop 状态的 Activity 时最先被系统考虑拿下来释放内存的。

④ 当一个 Activity 处于 pause 或 stop 的状态时，系统可以要求 Activity 结束（finish）或直接移除（kill）它。当它需要再度呈现在使用者面前时，它必须要能完整的重新启动及回复先前的状态。

### Activity 状态转换图





## Android 应用开发 4 使用 Service

### 什么是服务（Service）

服务是运行在后台的一段代码。它可以运行在它自己的进程，也可以运行在其他应用程序的上下文（context）里面，这取决于自身的需要。其他的组件可以绑定到一个服务（Service）上面，通过远程过程调用（RPC）来调用这个方法。例如：媒体播放器的服务，当用户退出媒体选择用户界面，仍然希望音乐可以继续播放，这时就是由服务（Service）来保证当用户界面关闭时音乐继续播放的。



### 如何使用服务

- ✓ 第一种是通过调用 Context.startService() 启动，调用 Context.stopService() 结束，startService() 可以传递参数给 Service。
- ✓ 第二种方式是通过调用 Context.bindService() 启动，调用 Context.unbindService() 结束，还可以通过 ServiceConnection 访问 Service。二者可以混合使用，比如说我可以先 startService() 再 unbindService()。

### Service 的生命周期

- ☹ startService() 后，即使调用 startService() 的进程结束了，Service 仍然存在，知道有进程调用 stopService()，或者 Service 自己自杀（stopSelf()）就没法了
- ☹ bindService() 后，Service 就和调用 bindService() 的进程同生共死，也就是说当调用 bindService() 的进程死了，那么它 bind 的 Service 也要跟着被结束，当然期间也可以调用 unbindService() 让 Service 结束
- ☹ 两种方式混合使用时，比如说你 startService() 了，我 bindService() 了，那么只有你 stopService() 了而且我也 unbindService() 了，这个 Service 才会被结束。

### 进程生命周期

- ♠ Android 系统将会尝试保留那些启动了的或者时绑定了的的服务进程
- ♠ 如果该服务正在进程的 onCreate(), onStart() 或者 onDestroy() 这些方法中执行时，那么主进程将会成为一个前台进程，以确保此代码不会被停止
- ♠ 如果服务已经开始，那么它的主进程会就重要性而言低于所有可见的进程但高于不可见的进程，由于只有少数几个进程是用户可见的，所以只要不是内存特别低，该服务不会停止。
- ♠ 如果有多个客户端绑定了服务，只要客户端中的一个对于用户是可见的，即认为该服务可见

### 使用服务进行音乐播放

Manifest.xml  
中的 Service  
定义

```
<service android:name=".Music">  
    <intent-filter>  
        <action android:name="com.liangshan.wuyong.START_AUDIO_SERVICE" />  
        <category android:name="android.intent.category.default" />  
    </intent-filter>  
</service>
```

Service 子类  
中的 Player

```
public void onStart(Intent intent, int startId) {  
    super.onStart(intent, startId);  
    player = MediaPlayer.create(this, R.raw.seven_days);  
    player.start();  
}  
  
public void onDestroy() {  
    super.onDestroy();  
    player.stop();  
}
```

Activity 中定  
义的 Intent  
开启相应的  
Service

```
startService(new Intent("com.liangshan.wuyong.START_AUDIO_SERVICE"));  
  
stopService(new Intent("com.liangshan.wuyong.START_AUDIO_SERVICE"));
```





# Android UI 布局

## Activity

- ◆ Android 应用程序基本功能单元
- ◆ 本身没有任何屏幕存在

## View 和 Viewgroup

- ◆ 表示在 Android 平台上的基本用户界面单元



## Views

- ◆ android.view.View
  - 为指定的屏幕矩形区域存储布局和内容
  - 处理尺寸和布局，绘制，焦点改变，翻页，按键、手势
  - widget 基类

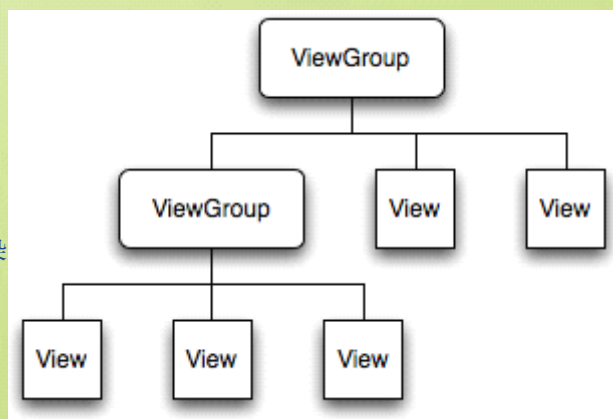
文本 TextView	输入框 EditText
输入法 InputMethod	活动方法 MovementMethod
按钮 Button	单选按钮 RadioButton
复选框 Checkbox	滚动视图 ScrollView

## Viewgroups

- ◆ android.view.ViewGroup
  - 包含并管理下级系列的 Views 和其他 ViewGroup
  - 布局的基类

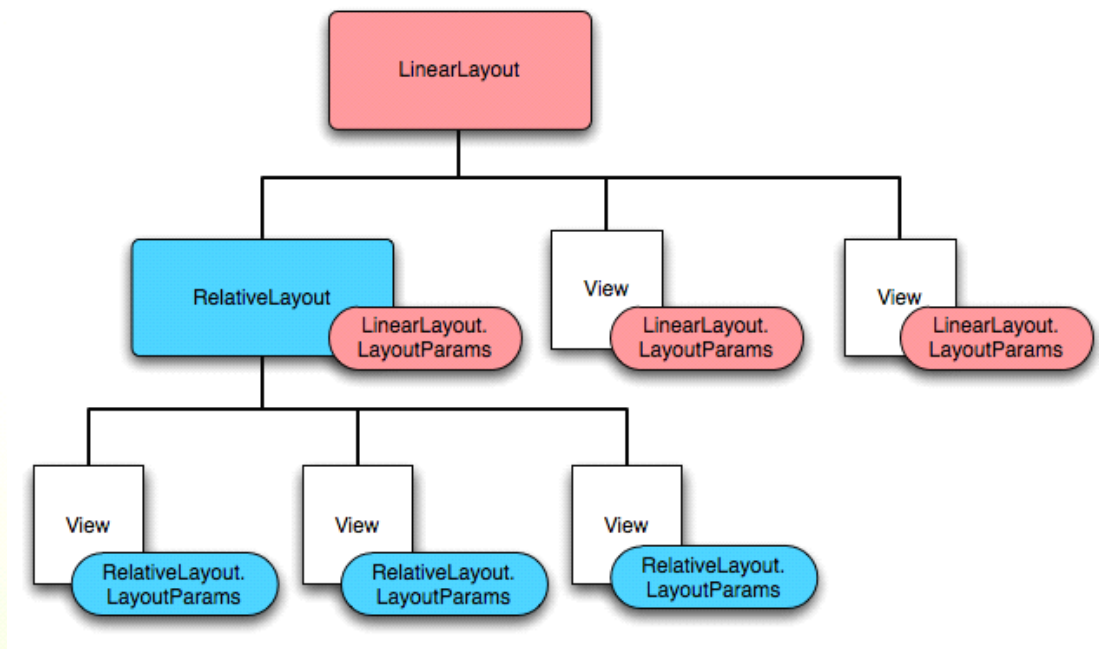
## UI 树状结构

- ◆ Android 中的 Activity
  - 定义使用一个 view 和 iewgroup 的树状节点
- ◆ setContentView() 方法
  - 被 Activity 调用来把树状节点连接到屏幕渲染



## LayoutParams (布局参数)

- ◆ 每一个 viewgroup 类使用一个继承于 ViewGroup.LayoutParams 的嵌套类
  - 包含定义了子节点 View 的尺寸和位置的属性类型



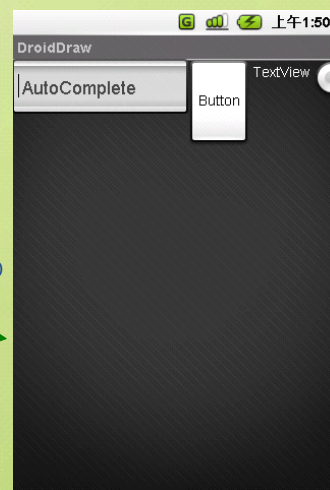
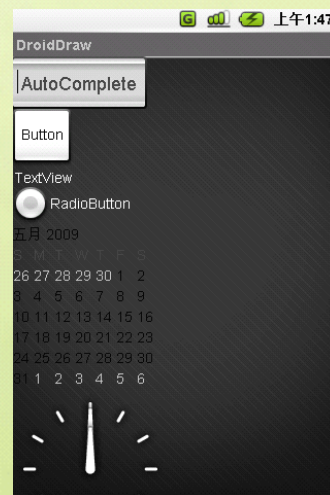
## 普通布局对象

### FrameLayout

- ◆ 最简单的布局对象
- ◆ 在屏幕上故意保留的空白空间，你可以之后填充一个单独的对象
  - 例如：一个你要更换的图片
- ◆ 所有子元素都钉到屏幕的左上角
- ◆ 不能为子元素指定位置

### LinearLayout

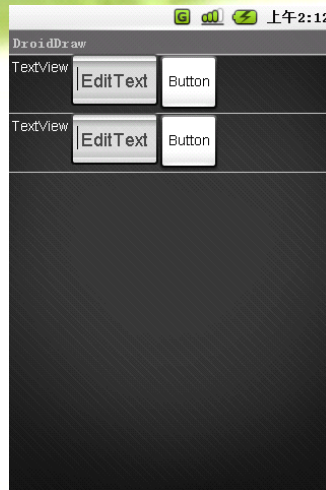
- ◆ 在一个方向上(垂直或水平)对齐所有子元素
  - 所有子元素一个跟一个地堆放
    - ✓ 一个垂直列表每行将只有一个子元素(无论它们有多宽)
    - ✓ 一个水平列表只是一列的高度(最高子元素的高度来填充)





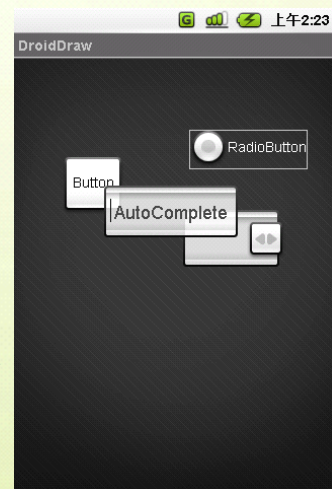
## TableLayout

- ◆ 把子元素放入到行与列中
- ◆ 不显示行、列或是单元格边界线
- ◆ 单元格不能横跨行，如 HTML 中一样



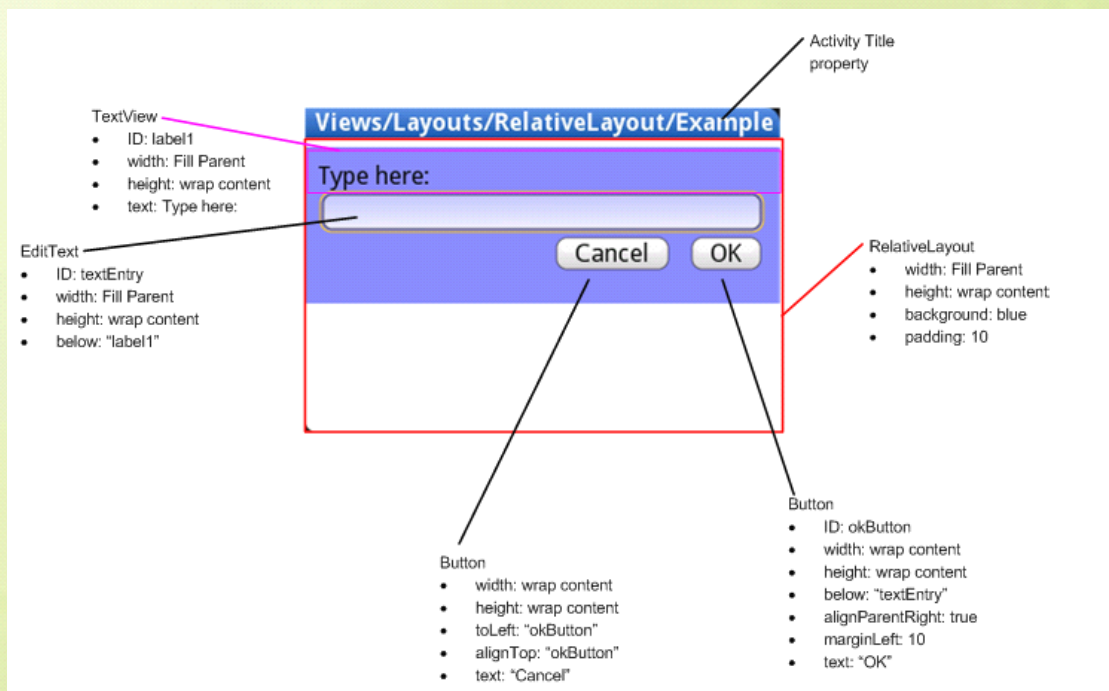
## AbsoluteLayout

- ◆ 使子元素能够指明确切的 X / Y 坐标显示在屏幕上
  - (0,0)是左上角
  - 当你下移或右移时，坐标值增加
- ◆ 允许元素重叠(但是不推荐)
- ◆ 注意：
  - 一般建议不使用 AbsoluteLayout 除非你有很好的理由来使用它
  - 因为它相当严格并且在不同的设备显示中不能很好地工作



## RelativeLayout

- ◆ 让子元素指定它们相对于其他元素的位置(通过 ID 来指定)或相对于父布局对象



### AndroidManifest.xml 中修改程序布局的 Theme 主题

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="zyf.GridViewTest"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:theme="@android:style/Theme.Light"
        android:label="@string/app_name">
        <activity android:name=".GridViewTest"
            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>
    <uses-sdk android:minSdkVersion="2" />
</manifest>
```





android  
developers



android