Bundle

对于Android开发者来说肯定非常眼熟，它经常出现在以下场合：

（1）Activity状态数据的保存与恢复涉及到的两个回调：void onSaveInstanceState (Bundle outState)、void onCreate (Bundle savedInstanceState)

（2）Fragment的setArguments方法：void setArguments (Bundle args)

（3）消息机制中的Message的setData方法：void setData (Bundle data)

Android的日志工具分为5个方法：

Log.v()-->verbose; Log.d()-->debug; Log.i()-->info; Log.w()-->warn; Log.e()-->error;

通常传入两个参数，第一个参数tag，用于对打印的信息进行过滤；

活动

（1）活动的创建

重写父类的onCreate()方法，调用父类的super.onCreate(savedInstanceState); setContentview(R.layout.first\_layout)方法用来设置布局。

（2）活动的状态

Android是使用任务来管理活动的，即返回栈。启动一个新活动会放在栈顶，当我们按下back键或者finish()方法去销毁一个活动时，处于栈顶的活动会出栈，系统总是会显示栈顶的活动给用户。

运行状态，暂停状态，停止状态，销毁状态

（3）活动的生命周期

onCreate();

onStart();

onResume() //这个方法在活动准备好和用户进行交互的时候调用，此时活动一定处于栈顶;

onPause()//在系统准备启动或回复另一个活动时调用

onStop()//活动完全不可见的时候调用

onDestory()//活动被销毁之前调用

onRestart()活动有之前的停止状态变回运行状态之前调用。

（4）活动回收之前的数据保存

onSaveInstanceState()该方法可以保证活动回收之前一定会被调用，将数据保存在Bundle中。Bundle.putString(“键”，“值”)保存字符串。重写该方法，然后在oncreate()函数中判断savedInstanceState 是否为空，之后取出数据。

（5）活动的启动模式

standard 不考虑活动是否已经在返回栈中存在

singleTop 发现返回栈的栈顶已经是该活动，则直接使用，不会再创建

singleTask 返回栈中只有一个实例，如果已经存在，则把这个活动之上的所有活动统统出栈。

singleInstance 一个活动一个栈

通过<activity>标签指定android:lanchMode属性来选择启动模式。

（6）getClass.getSimpleName()来获取当前所在的类名

（7）创建类管理器用户随时随地退出程序

添加控件的点击事件

（1）使用匿名内部类的方法

Button button1 = (Button) findViewById(R.id.button\_1);

button1.setOnClickListerer(new View.OnClickListerer(){

//重载onClick方法。

});

使用于只对某个控件添加点击事件。

（2）使用内部类的方法，实现View.OnClickListerer接口

（3）主类实现View.OnClickListerer接口

（4）在布局文件中为该View 设置 android: onClick属性，指定点击的方法名。

在活动中使用Toast

Toast.makeText(FirstActivity.this, "toast显示的内容", Toast.LENGTH\_SHORT).show;

在活动中使用Menu

需要先新建一个menu的xml文件，然后重写onCreateOptionMenu()方法，和对应的点击事件onOptionItemSelected()方法。

活动之间的跳转

1、使用显示的Intent

Intent itent = new Intent(FirsyActivity.this, SecondActivity.class);

startActivity(itent);

2、使用隐式Itent

需要先在AndroidManifest.xml中要跳转到的Activity中编辑新增一个条目<intent-filter>

<action android:name = >

<category android:name = >

</itent-filter>

然后在主Activity中：

Itent itent = new Itent(“action中的字符串”);

startActivity(itent);

Intent.setData()方法主要用于指定当前Intent正在操作的数据，而这些数据通常都是以字符串的形式传入到Uri.parse()方法中解析生成。

活动之间的数据传递

1、 向下一个活动中传递数据

在原活动中itent.putExtra(“键”，“对应得内容”),跳转到的活动中使用：

Itent itent = getIntent(); //获取用于启动活动的Itent

然后使用itent.getStringExtra(“键”)取出数据。

2、返回数据给上一个活动

使用startActivityForResult()启动活动，该方法可以在活动销毁的时候能够返回一个结果给上一个活动。需要两个参数，参数1为Intent,参数2为请求码，用于确认回调中数据的来源。

在跳转到的活动中：

Itent itent = new Intent();

Itent.putExtra(“键”，内容)；

SetResult(RESULT\_OK, itent)//第一个参数用于向上一个活动返回处理结果

跳转到的活动销毁之后，会回调到原活动的onActivityResult()方法，需要重写该方法。包含三个参数请求码，处理结果，带有数据的Itent.

ListView的简单用法

数组中的数据是无法直接传递给ListView，我们需要借用适配器来传递数据。适配器是数据和视图之间的桥梁。

ArrarAdapter<String> adapter = new ArrayAdapter<String>(ManiActivity.this, R.layout.simple\_list\_item\_1, data);

ListView listview = （Listview）findViewById(R.id.list\_view);

listview.setAdapter(adapter)

LayoutInflater的inflate（）方法接受三个参数，1（资源id）,2(ViewGroup parent), 3(false);

优化使用内部类ViewHolder存储控件的实例。

ListView 添加点击事件

listView.setOnItemClickListener(new AdapterView.OnItemClickListener(){

@Override

public void onItemClick(){

});

广播

标准广播，有序广播