**Android生命周期的学习**

1. Activity生命周期的4中状态和7个方法

生命周期为： 1.完整onCreate() 只会调用一次，在创建时，进行初始化。当此前的Activity由于内存不够被杀死的话。返回时，还是会调用onCreate()

,onStart() Activity处于可见状态

,onResume() Activity获得焦点

,onPause 暂停状态，失去焦点

,onStop 不可见状态

,onDestory 销毁状态

2.可见生命周期，有一个活动区启动另一个活动、点击home建，点击menu返回后台。由onStart() 到 onStop,当活动返回时，调用 onRestart() onStart() onResume()。

3,前台生命周期，弹出一个dialog,后面的活动还是可见，只是失去焦点。onResume() ,onPasue()，dialog消失时调用，onResume()方法。

补充：超级完整周期：onCreate() ,一般都是在create中进行恢复数据

,onStart(),

onRestoreInstanceState()，此方法调用只有活动被系统杀死时才可以调用恢复，如旋转屏幕，内存不够杀死。

onResume(),

onAttachedToWindow(),活动添加到窗口

onPause(),

onSaveInstanceState()，在活动变得不可见时，调用此方法保存活动的状态，以便恢复

onStop(),

onDestory()

onDetachedFromWindow(),活动从窗口中移除

onRestart()

1. Android的框架层：由上向下：应用层 ：包含home，contacts，phone，等程序

应用框架层：有activitiy Manager,window Manager等

系统运行库层：一些C/C++库，android运行库和虚拟机

Linux内核层：蓝牙驱动，显示驱动，wifi等为上层提供硬件支持。

**Android的四大组件**

**Activity:**

1. Activity:所有可见的都是Activity。
2. 任务栈：activitiy每启动一个，就会添加到任务栈中。每一个可见的activity都处于栈顶，当销毁一个活动时，activity会从栈顶移除，出栈。
3. Activity的四种启动模式：

1.standard（默认）标准模式，使用startActivity（intent）启动一次Activity，就会创建一个实例存储在任务栈中，然后返回时，会有多个重复的Activity。

2.singleTop模式，如果此时activity处于栈顶时，那么跳转到这个activity时不会再次创建一个新的实例，而是使用这个实例。

3.singleTask模式，跳转Activity时，会现在栈中查看有无此活动的实例，如果有的话，则将此activity之上的所有活动全部出栈，此时此活动处于栈顶，跳转完成。反之，创建此activity实例。

4.singleInstance模式，也是系统单例模式。如果此activity模式是singleInstance，则在启动一个活动时，会创建一个task栈存放新创建的活动。此模式下一个栈中，只允许一个activity存在。

1. 在Activity之外进行页面跳转的时候，Intent需要添加FLAG,标记，如intent. addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK)。 （因为activity中以默认加上）
2. Intent:意图。启动方式，1.component，显示跳转。

2.action，category行为和类别配合使用。隐式跳转

3.data，type.类别和数据，跳转时可以携带，打开网页，打电话等。

6) Activity隐式调用时，必须要写<category android:name="android.intent.category.DEFAULT" />不添加的话就会报错。

**Service:**

1. Service：后台进行的一切不可见，不用用户知道的操作，如下载，音乐播放等。
2. Service两种启动方式：

1.通过startService(),启动，stopService()关闭，onCreate,onStartCommand,onDestory.

2.通过bindService(),启动,unbindService关闭，onCreate,onbind,onUnbind,onDestory

1. Android 5.0时启动service必须要显示调用，想要隐式调用必须指明包名。
2. Service与线程的区别，service是不可见的但是也运行在主线程，不能直接在service中作耗时操作，不然还是会导致ANR(application not responding).所以正确的做法是在service中开一个线程来进行耗时操作。
3. 为什么不在activity中做耗时操作呢？因为activity要与用户交互频繁，一旦被销毁，就不能对线程进行控制，而service可以与多有的activity进行交互，可以很方便的进行控制。
4. 进程分为：前台进程，可见进程，服务进程，后台进程，空进程。
5. 前台service：因为service的优先级较低，当内存不足时，有可能会被杀死的。所以要将service变成前台service.,使用NotificationCompat.Bulider 来创建信息，向下兼容，使用build()生成Notification对象。然后使用NotificationManager对象的notify方法来显示通知，在使用startForeground()方法来设置为前台服务。
6. 远程服务：在androidManifest.xml中的service设置process=”:remote”表名为远程服务，不在一个进程中。或者创建AIDL，实现进程间通信。

**BroadcastReceiver：**

1. 静态注册：在程序没有启动的情况下，就能接收到广播信息，在AndroidManifest中注册，然后在代码中继承BroadcastReceiver类重写onReceive方法
2. 动态注册：只有程序运行时才能接受到广播，较安全，也是要实现一个接收器，在代码中registerReceiver，unregister..注册和注销。使用sendBroadcast发送广播
3. 广播分类：可分为有序广播和无序广播。有序广播按照优先级进行发送一个一个接收，可以在manifest中设置优先级-1000~1000之间。无序广播，都可以接受到信息。
4. 广播的截断：只有有序广播可以被截断，使用方法abortBroadcast()
5. 系统广播：系统中默认的广播信息，如开机启动，电池电量，网络改变。
6. NFC：近距离无线通信，1.在AndroidManifest中声明nfc权限。2.在获取nfc的activity中获取注册action信息

**ContentProvider:**

1. 主要用来提供数据的共享，实现一个应用对另一个应用的增删改查。
2. 继承ContentProvider，在AndroidManifest中注册，填写authority=包名+类名
3. 在ContentProvider中初始化UriMatcher,添加数据。
4. 重写相应的增删改查方法。
5. MimeType类型返回为，vnd.android.cursor.dir/person，多条数据 ，vnd.android.cursor.item/person，单条数据返回
6. 在activity查询时，获取contentResolve对象，调用query方法，传入uri
7. Uri为content://包名/类名/数据库列

**Android中的MVVM与DataBinding**

1. Build Environment:在app的module中bulid.gradle配置dataBinding
2. 布局中根布局为layout，data元素里面有两个属性 1.variable 变量名用来绑定数据

2.type类型名表示绑定的类，在接下来的控件中进行绑定如：textView的text=”@{…}”

1. 然后在activity中，会自动生成刚才binding的layout布局,转换成为pascal以Binding结尾，如activity\_main ---》 ActivityMainBinding amb
2. 然后调用setcontent()进行绑定，调用amb.setVarible()。进行绑定
3. 事件处理两种方式：1.方法引用：如onclick=“test”,一个重要的好处是表达式在编译期间就被确定了。

2.监听器绑定： lambda表达式

1. Import，就像在java代码中一样，dataBinding在xml的data元素下也可以进行导包，因为java.lang包默认导入故可以直接使用。
2. 类名重复时，可以添加alias标签
3. 导入的类型，也可以用于variable元素中
4. 可以在xml中直接使用带id的view，应用时采用驼峰命名法，系统自动生成为public final TextView t; 比findviewbyId速度快
5. Bingding类会自动生成一个context对象，也就是rootView的getContext()的返回值
6. binding类名自动生成，可以通过修改在data元素下的class标签
7. dataBinding支持include，但是不能再merge节点中使用，在布局中使用。
8. 常用表达式，1.数学表达式，2字符串拼接+，3逻辑运算&&||，4一元++，--，！，~，

5．比较运算，6.instanceOf，7.方法调用，8.类型转换

9.数组存取，10,.三元运算符?;;

1. 在使用上述表达式时，注意转义字符的使用，如<--&lt, >-- &gt
2. 不支持的操作符为this，super，new，泛型调用
3. Null合并运算??，在非null的时候选择左边，反之右边。User.lastName??user.firstName

**Android的基本UI学习**

1. Android的布局分为，四大布局：

1.FrameLayout：最简单的布局管理器，默认view放在左上角，视图会层叠排放。

2.LinearLayout:线性布局，分为水平和垂直

3.RelativeLayout:相对布局，以父布局或者子view的位置，进行布局

4.GridLayout:网格布局， 在android4.0中引用，很灵活，简化布局，减少嵌套

1. 布局的优化：多层嵌套可对应用程序性能和响应能力造成影响。
2. 嵌套最好控制在10层以内，最好使用merge和include结合。
3. 布局中的view不能超过80个
4. 使用Lint工具来分析布局：android studio中已经集成了lint工具
5. Merge标签可以减少视图间的多余，如主布局为垂直布局，include的布局也是垂直布局，那么我们可以再include的布局使用merge标签，引用时自动优化。
6. ViewStub标签，好处是当你需要时才会加载。不常用的布局如进度条，显示错误信息等可以使用，viewstub是个不可见的，大小为0的view。想要使用时，可以findviewbyid在setVisibility,或者直接find。.inflate().

具体的使用：1.在布局中使用layout：指向特定的布局，inflateid：特定的view

2.在代码中显示使用inflate或者setVisibility

1. Fragment：1.fragment的生命周期
2. Fragment添加到用户界面的两种方式：

1.静态加载，直接在布局中引用，指定name元素，每个fragment都必须使用唯一的标示符，id或者tag.

2.动态加载，在程序中加载，使用fragmentTransaction的add,replace,hide,show方法

1. FragmentManager的管理使用，在提交之前调用addToBackStack(null)返回栈
2. Commit()方法不会立即提交，而是异步操作，也可以使用exectuPendingTransactions()来立即执行commit方法。
3. Fragment与Activity的通信，在fragment中使用getActivity().findViewById.在activity中可以getFragmentManager.findFragmentByid/tag
4. 在fragment中使用菜单，要现在onCreat中调用setHasOptionsMenu(true)
5. 使用popBackStack()可以使当前fragmentManager中的fragment弹栈。
6. ListFragment：一个自带listview，和点击事件的fragment，布局包含listview，id为@id/android:list指定，
7. DialogFragment的使用：好处：屏幕旋转时的数据保存，可以直接使用AlertDialog和dialog重组

**TextView:**

1. textStyle:normal,bold(加粗)，italic(斜体)
2. lineSpacingExtra,行间距 ，lineSpacingMultiplier 行间距的倍数
3. shadowColor设置阴影颜色，
4. shadowRadius设置阴影模糊程度，设置0.1为字体颜色，建议设置3.0
5. shadowDx水平方向的偏移
6. shadowDy竖直方向的偏移
7. drawableXXX，（top，bottom。。）可以给textView设置不同方向的图片
8. autoLink,（all,emil,phone,）点击可以跳转
9. 可以使用Html.fromHtml来转换超文本
10. SpannableString & SpannableStringBuilder定制文本
11. EditText使用spannableString来输入图片
12. Button的状态的改变，stateListDrawable，使用selector标签设置选中状态，shape修改颜色。

**ListView和Adapter:**

1. Adapter:BaseAdapter的使用，ArrayAdapter,simpleDateAdapter
2. Adapter的优化：复用布局以及创建viewHolder.。adapter支持复用机制
3. Listview和button焦点抢占问题，解决问题给button设置focusable为false
4. 问题？切换fragment导致，listview数据重复？因为list集合重复添加了数据，在mainactivity中一直持有Fragment的引用，故list集合也就存活，而在viewCreate中初始化数据，导致会重复添加数据
5. ExpandableListView的简单使用以及BaseExpandableListAdapter
6. viewFlipper实现图片轮播图的两种方式。Xml中定义，代码中addView（）
7. 自定义重写onTouchEvent和手势实现ViewFlipper的滑动
8. GestureDetector手势的使用：1.onDown 2.onFling 快速移动，并松开 3.onScroll 4.onShowPress

5.onSingTabUp 手指离开触摸屏的一刹那

1. 如何在fragment中使用onTouchEvent？使用接口回调的方式在activity中进行事件的分发dispatchTouchEvent()中进行事件分发

**Android的数据存储**

1. 数据