Android 常问基础知识点

1、四大组件是什么

1）Activity：用户可操作的可视化界面，为用户提供一个完成操作指令的窗口。一个 Activity 通常是一个单独的屏幕，Activity 通过 Intent 来进行通信。Android 中会维持一个 Activity Stack，当一个新 Activity 创建时，它就会放到栈顶，这个 Activity 就处于运行状态。

2）Service：服务，运行在手机后台，适合执行不需和用户交互且还需长期运行的任务。

3）ContentProvider：内容提供者，使一个应用程序的指定数据集提供给其他应用程序，其他应用可通过 ContentResolver 类从该内容提供者中获取或存入数据。它提供了一种跨进程数据共享的方式，当数据被修改后，ContentResolver 接口的 notifyChange 函数通知那些注册监控特定 URI 的 ContentObserver 对象。

如果 ContentProvider 和调用者在同一进程中，ContentProvider 的方法(query/insert/update/delete 等)和调用者在同一线程中；如果 ContentProvider 和调用者不在同一进程，ContentProvider 方法会运行在它自身进程的一个 Binder 线程中。

4）Broadcast Receiver: 广播接收者，运用在应用程序间传输信息，可以使用广播接收器来让应用对一个外部事件做出响应。

2、四大组件的生命周期和简单用法

1）Activity：onCreate()->onStart()->onResume()->onPause()->onStop()->onDestory()

onCreate()：为 Activity 设置布局，此时界面还不可见；

onStart(): Activity 可见但还不能与用户交互，不能获得焦点

onRestart(): 重新启动 Activity 时被回调

onResume(): Activity 可见且可与用户进行交互

onPause(): 当前 Activity 暂停，不可与用户交互，但还可见。在新 Activity 启动前被系统调用保存现有的 Activity 中的持久数据、停止动画等。

onStop(): 当 Activity 被新的 Activity 覆盖不可见时被系统调用

onDestory(): 当 Activity 被系统销毁杀掉或是由于内存不足时调用

2）Service

a) onBind 方式绑定的：onCreate->onBind->onUnBind->onDestory（不管调用 bindService 几次，

onCreate 只会调用一次，onStart 不会被调用，建立连接后，service 会一直运行，直到调用 unBindService 或是之前调用的 bindService 的 Context 不存在了，系统会自动停止 Service,对应的 onDestory 会被调用）

b) startService 启动的：onCreate->onStartCommand->onDestory(start 多次，onCreate 只会被调用一次，onStart 会调用多次，该 service 会在后台运行，直至被调用 stopService 或是 stopSelf)

c) 又被启动又被绑定的服务，不管如何调用 onCreate()只被调用一次，startService 调用多少次，onStart 就会被调用多少次，而 unbindService 不会停止服务，必须调用 stopService 或是 stopSelf 来停止服务。必须 unbindService 和 stopService(stopSelf）同时都调用了才会停止服务。

3）BroadcastReceiver

a) 动态注册：存活周期是在 Context.registerReceiver 和 Context.unregisterReceiver 之间，BroadcastReceiver 每次收到广播都是使用注册传入的对象处理的。

b) 静态注册：进程在的情况下，receiver 会正常收到广播，调用 onReceive 方法；生命周期只存活在 onReceive 函数中，此方法结束，BroadcastReceiver 就销毁了。onReceive()只有十几秒存活时间，在 onReceive()内操作超过 10S，就会报 ANR。

进程不存在的情况，广播相应的进程会被拉活，Application.onCreate 会被调用，再调用 onReceive。

4）ContentProvider：应该和应用的生命周期一样，它属于系统应用，应用启动时，它会跟着初始化，应用关闭或被杀，它会跟着结束。

3、Activity 之间的通信方式

1）通过 Intent 方式传递参数跳转

2）通过广播方式

3）通过接口回调方式

4）借助类的静态变量或全局变量

5）借助 SharedPreference 或是外部存储，如数据库或本地文件

4、Activity 各种情况下的生命周期

1) 两个 Activity(A->B) 切换 (B 正常的 Activity) 的生命周期：onPause(A)->onCreate(B)->onStart(B)->onResume(B)->oStop(A)

这时如果按回退键回退到 A onPause(B)->onRestart(A)->onStart(A)->onResume(A)->oStop(B)

如果在切换到 B 后调用了 A.finish()，则会走到 onDestory(A)，这时点回退键会退出应用

2) 两个 Activity(A->B)切换(B 透明主题的 Activity 或是 Dialog 风格的 Acivity)的生命周期：onPause(A)->onCreate(B)->onStart(B)->onResume(B)

这时如果回退到 A onPause(B)->onResume(A)->oStop(B)->onDestory(B)

3) Activity(A) 启动后点击 Home 键再回到应用的生命周期：onPause(A)->oStop(A)->onRestart(A)->onStart(A)->onResume(A)

5、横竖屏切换的时候，Activity 各种情况下的生命周期

1）切换横屏时：onSaveInstanceState->onPause->onStop->onDestory->onCreate->onStart->onRestoreInstanceState->onResume

2) 切换竖屏时：会打印两次相同的 log

onSaveInstanceState->onPause->onStop->onDestory->onCreate->onStart->onRestoreInstanceState->onResume->onSaveInstanceState->onPause->onStop->onDestory->onCreate->onStart->onRestoreInstanceState->onResume

3) 如果在 AndroidMainfest.xml 中修改该 Activity 的属性，添加 android:configChanges="orientation"

横竖屏切换，打印的 log 一样，同 1)

4) 如果 AndroidMainfest.xml 中该 Activity 中的 android:configChanges="orientation|keyboardHidden"，则只会打印

onConfigurationChanged->

6、Activity 与 Fragment 之间生命周期比较

Fragment 生命周期：onAttach->onCreate->onCreateView->onActivityCreated->onStart->onResume->onPause->onStop->onDestoryView->onDestory->onDetach

切换到该 Fragment：onAttach->onCreate->onCreateView->onActivityCreated->onStart->onResume

按下 Power 键：onPause->onSaveInstanceState->onStop

点亮屏幕解锁：onStart->onRestoreInstanceState->onResume

切换到其他 Fragment: onPause->onStop->onDestoryView

切回到该 Fragment: onCreateView->onActivityCreated->onStart->onResume

退出应用：onPause->onStop->onDestoryView->onDestory->onDetach

7、Activity 上有 Dialog 的时候按 Home 键时的生命周期

AlertDialog 并不会影响 Activity 的生命周期，按 Home 键后才会使 Activity 走 onPause->onStop，AlertDialog 只是一个组件，并不会使 Activity 进入后台。

8、两个 Activity 之间跳转时必然会执行的是哪几个方法？

前一个 Activity 的 onPause，后一个 Activity 的 onResume

9、前台切换到后台，然后再回到前台，Activity 生命周期回调方法。弹出 Dialog，生命值周期回调方法。

1）前台切换到后台，会执行 onPause->onStop，再回到前台，会执行 onRestart->onStart->onResume

2) 弹出 Dialog，并不会影响 Activity 生命周期

10、Activity 的四种启动模式对比

1）standard：标准启动模式（默认），每启动一次 Activity，都会创建一个实例，即使从 ActivityA startActivity ActivityA,也会再次创建 A 的实例放于栈顶，当回退时，回到上一个 ActivityA 的实例。

2) singleTop：栈顶复用模式，每次启动 Activity，如果待启动的 Activity 位于栈顶，则不会重新创建 Activity 的实例，即不会走 onCreate->onStart，会直接进入 Activity 的 onPause->onNewIntent->onResume 方法

3) singleInstance: 单一实例模式，整个手机操作系统里只有一个该 Activity 实例存在，没有其他 Actvity,后续请求均不会创建新的 Activity。若 task 中存在实例，执行实例的 onNewIntent()。应用场景：闹钟、浏览器、电话

4) singleTask：栈内复用，启动的 Activity 如果在指定的 taskAffinity 的 task 栈中存在相应的实例，则会把它上面的 Activity 都出栈，直到当前 Activity 实例位于栈顶，执行相应的 onNewIntent()方法。如果指定的 task 不存在，创建指定的 taskAffinity 的 task,taskAffinity 的作用，进入指写 taskAffinity 的 task,如果指定的 task 存在，将 task 移到前台，如果指定的 task 不存在，创建指定的 taskAffinity 的 task. 应用场景：应用的主页面

11、Activity 状态保存于恢复

Activity 被主动回收时，如按下 Back 键，系统不会保存它的状态，只有被动回收时，虽然这个 Activity 实例已被销毁，但系统在新建一个 Activity 实例时，会带上先前被回收 Activity 的信息。在当前 Activity 被销毁前调用 onSaveInstanceState(onPause 和 onStop 之间保存)，重新创建 Activity 后会在 onCreate 后调用 onRestoreInstanceState（onStart 和 onResume 之间被调用），它们的参数 Bundle 用来数据保存和读取的。

保存 View 状态有两个前提：View 的子类必须实现了 onSaveInstanceState; 必须要设定 Id，这个 ID 作为 Bundle 的 Key

12、fragment 各种情况下的生命周期

正常情况下的生命周期：onAttach->onCreate->onCreateView->onActivityCreated->onStart->onResume->onPause->onStop->onDestoryView->onDestory->onDetach

1)Fragment 在 Activity 中 replace onPause(旧)->onAttach->onCreate->onCreateView->onActivityCreated->onStart->onResume->onStop(旧)->onDestoryView(旧)

如果添加到 backStack 中，调用 remove()方法 fragment 的方法会走到 onDestoryView，但不会执行 onDetach()，即 fragment 本身的实例是存在的，成员变量也存在，但是 view 被销毁了。如果新替换的 Fragment 已在 BackStack 中，则不会执行 onAttach->onCreate

13、Fragment 状态保存 onSaveInstanceState 是哪个类的方法，在什么情况下使用？

在对应的 FragmentActivity.onSaveInstanceState 方法会调用 FragmentController.saveAllState，其中会对 mActive 中各个 Fragment 的实例状态和 View 状态分别进行保存.当 Activity 在做状态保存和恢复的时候, 在它其中的 fragment 自然也需要做状态保存和恢复.

14、Fragment.startActivityForResult 是和 FragmentActivity 的 startActivityForResult？

如果希望在 Fragment 的 onActivityResult 接收数据，就要调用 Fragment.startActivityForResult，而不是 Fragment.getActivity().startActivityForResult 。Fragment.startActivityForResult->FragmentActivitymHost.HostCallbacks.onStartActivityFromFragment->FragmentActivity.startActivityFromFragment 。如果 request=-1 则直接调用 FragmentActivity.startActivityForResult，它会重新计算 requestCode，使其大于 0xfffff。

15、如何实现 Fragment 的滑动？

ViewPager+FragmentPagerAdapter+List<Fragment>

16、fragment 之间传递数据的方式？

1)在相应的 fragment 中编写方法，在需要回调的 fragment 里获取对应的 Fragment 实例，调用相应的方法；

2）采用接口回调的方式进行数据传递；

a) 在 Fragment1 中创建一个接口及接口对应的 set 方法; b) 在 Fragment1 中调用接口的方法；c)在 Fragment2 中实现该接口；

3）利用第三方开源框架 EventBus

17、service 和 activity 怎么进行数据交互？

1）通过 bindService 启动服务，可以在 ServiceConnection 的 onServiceConnected 中获取到 Service 的实例，这样就可以调用 service 的方法，如果 service 想调用 activity 的方法，可以在 service 中定义接口类及相应的 set 方法，在 activity 中实现相应的接口，这样 service 就可以回调接口言法；

2）通过广播方式

18、说说 ContentProvider、ContentResolver、ContentObserver 之间的关系

ContentProvider 实现各个应用程序间数据共享，用来提供内容给别的应用操作。如联系人应用中就使用了 ContentProvider，可以在自己应用中读取和修改联系人信息，不过需要获取相应的权限。它也只是一个中间件，真正的数据源是文件或 SQLite 等。

ContentResolver 内容解析者，用于获取内容提供者提供的数据，通过 ContentResolver.notifyChange(uri)发出消息

ContentObserver 内容监听者，可以监听数据的改变状态，观察特定 Uri 引起的数据库变化，继而做一些相应的处理，类似于数据库中的触发器，当 ContentObserver 所观察的 Uri 发生变化时，便会触发它。

19、请描述一下广播 BroadcastReceiver 的理解

BroadcastReceiver 是一种全局监听器，用来实现系统中不同组件之间的通信。有时候也会用来作为传输少量而且发送频率低的数据，但是如果数据的发送频率比较高或者数量比较大就不建议用广播接收者来接收了，因为这样的效率很不好，因为 BroadcastReceiver 接收数据的开销还是比较大的。

20、广播的分类

1）普通广播：完全异步的，可以在同一时刻（逻辑上）被所有接收者接收到，消息传递的效率比较高，并且无法中断广播的传播。

2）有序广播：发送有序广播后，广播接收者将按预先声明的优先级依次接收 Broadcast。优先级高的优先接收到广播，而在其 onReceiver()执行过程中，广播不会传播到下一个接收者，此时当前的广播接收者可以 abortBroadcast()来终止广播继续向下传播，也可以将 intent 中的数据进行修改设置，然后将其传播到下一个广播接收者。 sendOrderedBroadcast(intent, null);//发送有序广播

3）粘性广播：sendStickyBroadcast()来发送该类型的广播信息，这种的广播的最大特点是，当粘性广播发送后，最后的一个粘性广播会滞留在操作系统中。如果在粘性广播发送后的一段时间里，如果有新的符合广播的动态注册的广播接收者注册，将会收到这个广播消息，虽然这个广播是在广播接收者注册之前发送的，另外一点，对于静态注册的广播接收者来说，这个等同于普通广播。

21、广播使用的方式和场景

1）App 全局监听：在 AndroidManifest 中静态注册的广播接收器，一般我们在收到该消息后，需要做一些相应的动作，而这些动作与当前 App 的组件，比如 Activity 或者 Service 的是否运行无关，比如我们在集成第三方 Push SDK 时，一般都会添加一个静态注册的 BroadcastReceiver 来监听 Push 消息，当有 Push 消息过来时，会在后台做一些网络请求或者发送通知等等。

2）组件局部监听：这种主要是在 Activity 或者 Service 中使用 registerReceiver()动态注册的广播接收器，因为当我们收到一些特定的消息，比如网络连接发生变化时，我们可能需要在当前 Activity 页面给用户一些 UI 上的提示，或者将 Service 中的网络请求任务暂停。所以这种动态注册的广播接收器适合特定组件的特定消息处理。

22、在 manifest 和代码中如何注册和使用 BroadcastReceiver?

1）mainfest 中注册:静态注册的广播接收者就是一个常驻在系统中的全局监听器，也就是说如果你应用中配置了一个静态的 BroadcastReceiver，而且你安装了应用而无论应用是否处于运行状态，广播接收者都是已经常驻在系统中了。

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="com.smilexie.test.intent.mybroadcastreceiver"/>
    </intent-filter>
</receiver>
```

2) 动态注册:动态注册的广播接收者只有执行了 registerReceiver(receiver, filter)才会开始监听广播消息，并对广播消息作为相应的处理。

```
IntentFilter fiter = new IntentFilter("com.smilexie.test.intent.mybroadcastreceiver");
MyBroadcastReceiver receiver = new MyBroadcastReceiver();
registerReceiver(receiver, filter);

//撤销广播接受者的动态注册
unregisterReceiver(receiver);
```

23、本地广播和全局广播有什么差别？

1）LocalBroadcastReceiver 仅在自己的应用内发送接收广播，也就是只有自己的应用能收到，数据更加安全。广播只在这个程序里，而且效率更高。只能动态注册，在发送和注册的时候采用 LocalBroadcastManager 的 sendBroadcast 方法和 registerReceiver 方法。

2）全局广播：发送的广播事件可被其他应用程序获取，也能响应其他应用程序发送的广播事件（可以通过 exported－是否监听其他应用程序发送的广播 在清单文件中控制） 全局广播既可以动态注册，也可以静态注册。

24、AlertDialog,popupWindow,Activity 区别

（1）Popupwindow 在显示之前一定要设置宽高，Dialog 无此限制。

（2）Popupwindow 默认不会响应物理键盘的 back，除非显示设置了 popup.setFocusable(true);而在点击 back 的时候，Dialog 会消失。

（3）Popupwindow 不会给页面其他的部分添加蒙层，而 Dialog 会。

（4）Popupwindow 没有标题，Dialog 默认有标题，可以通过 dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);取消标题

（5）二者显示的时候都要设置 Gravity。如果不设置，Dialog 默认是 Gravity.CENTER。

（6）二者都有默认的背景，都可以通过 setBackgroundDrawable(new ColorDrawable(android.R.color.transparent));去掉。
（7）Popupwindow 弹出后，取得了用户操作的响应处理权限，使得其他 UI 控件不被触发。而 AlertDialog 弹出后，点击背景，AlertDialog 会消失。

25、Application 和 Activity 的 Context 对象的区别
1）Application Context 是伴随应用生命周期；不可以 showDialog, startActivity, LayoutInflation

可以 startService\BindService\sendBroadcast\registerBroadcast\load Resource values

2）Activity Context 指生命周期只与当前 Activity 有关，而 Activity Context 这些操作都可以，即凡是跟 UI 相关的，都得用 Activity 做为 Context 来处理。

一个应用 Context 的数量=Activity 数量+Service 数量+1（Application 数量）

--Android 属性动画特性
--如何导入外部数据库?

--LinearLayout、RelativeLayout、FrameLayout 的特性及对比，并介绍使用场景。
https://www.jianshu.com/p/2f33c29459b8

--谈谈对接口与回调的理解
https://www.jianshu.com/p/1cc11d19635e

--回调的原理
https://www.cnblogs.com/xrq730/p/6424471.html
--写一个回调 demo

--介绍下 SurfView
https://blog.csdn.net/luoshengyang/article/details/8661317/
https://blog.csdn.net/android_cmos/article/details/68955134

--RecycleView 的使用

--序列化的作用，以及 Android 两种序列化的区别

--差值器

https://www.cnblogs.com/mengdd/p/3346003.html

https://blog.csdn.net/pzm1993/article/details/77926373

实现 Interpolator 接口，设置值的变化趋势，SDK 中包含了匀速插值器、加速插值器、减速插值器、先加速再减速、弹

--估值器

实现 TypeEvaluatior 接口

https://blog.csdn.net/l_wwbs/article/details/53388924

--Android 中数据存储方式

https://www.cnblogs.com/blosaa/p/6244050.html