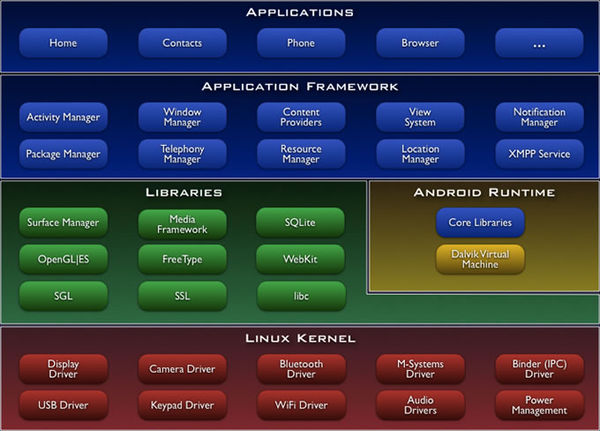
Android基础

一、Android的系统架构 由4部分组成(最初就是学习第一层,然后学习借用第二层的工具)

Android的系统架构和其[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)一样，采用了分层的架构。Android分为四个层，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和Linux内核层。  
  
一、应用程序  
Android会同一系列核心应用程序包一起发布，该应用程序包包括客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用[JAVA语言](https://www.baidu.com/s?wd=JAVA%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)编写的。  
  
二、应用程序框架  
开发人员也可以完全访问核心应用程序所使用的API框架。该应用程序的[架构设计](https://www.baidu.com/s?wd=%E6%9E%B6%E6%9E%84%E8%AE%BE%E8%AE%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)简化了组件的重用;任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块（不过得遵循框架的安全性）。同样，该应用程序重用机制也使用户可以方便的替换程序组件。  
  
三、系统运行库  
Android 包含一些[C/C++](https://www.baidu.com/s?wd=C/C++&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)库，这些库能被Android系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。  
  
四、Linux内核层  
Android内核是基于Linux 内核的修改的[内核版本](https://www.baidu.com/s?wd=%E5%86%85%E6%A0%B8%E7%89%88%E6%9C%AC&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)，它提供了用于支持Android平台的设备驱动。

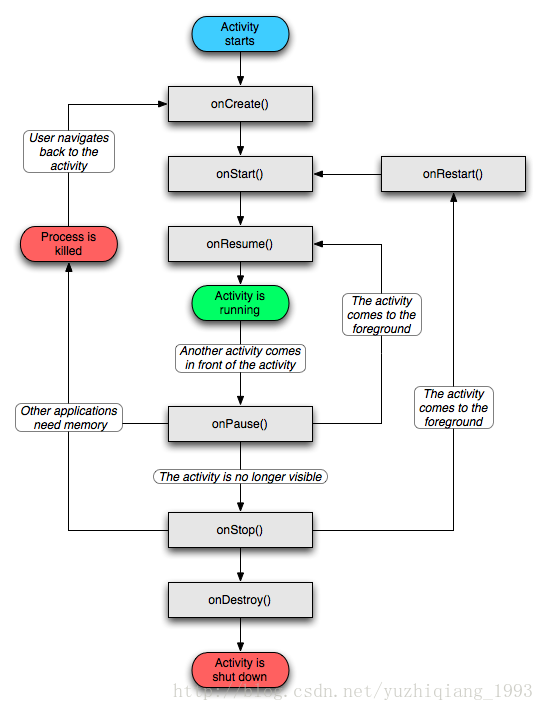


# 二、Android应用程序由Android四大核心组件组成（****Activity,Service,ContentProvider,BroadCastReceiver****）

1、Activity

Activity的中文意思是活动，代表手机屏幕的一屏，或是平板电脑中的一个窗口， 提供了和用户交互的可视化界面。Activity是用于处理UI相关业务的，比如加载界面、 监听用户操作事件。

* + 1. 生命周期 就是从开始到结束的执行流程



1.Activity实例是由系统自动创建，并在不同的状态期间回调相应的方法。一个最简单的完整的Activity生命周期会按照如下顺序回调：onCreate -> onStart -> onResume -> onPause -> onStop -> onDestroy。称之为**entire lifetime。**

2.当执行onStart回调方法时，Activity开始被用户所见（也就是说，onCreate时用户是看不到此Activity的，那用户看到的是哪个？当然是此Activity之前的那个Activity），一直到onStop之前，此阶段Activity都是被用户可见，称之为**visible lifetime。**

3.当执行到onResume回调方法时，Activity可以响应用户交互，一直到onPause方法之前，此阶段Activity称之为**foreground lifetime。**

在实际应用场景中，假设A Activity位于栈顶，此时用户操作，从A Activity跳转到B Activity。那么对AB来说，具体会回调哪些生命周期中的方法呢？回调方法的具体回调顺序又是怎么样的呢？

开始时，A被实例化，执行的回调有A:onCreate -> A:onStart -> A:onResume。

**当用户点击A中按钮来到B时，假设B全部遮挡住了A，将依次执行A:onPause -> B:onCreate -> B:onStart -> B:onResume -> A:onStop。**

**此时如果点击Back键，将依次执行B:onPause -> A:onRestart -> A:onStart -> A:onResume -> B:onStop -> B:onDestroy。**

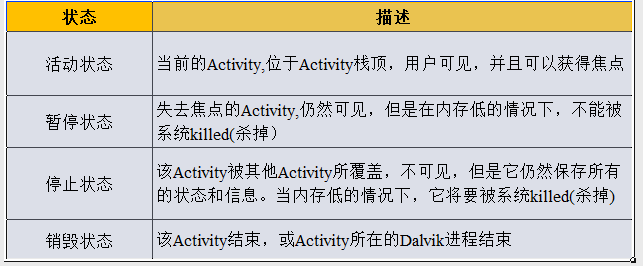
至此，Activity栈中只有A。在Android中，有两个按键在影响Activity生命周期这块需要格外区分下，即Back键和Home键。我们先直接看下实验结果：

**此时如果按下Back键，系统返回到桌面，并依次执行A:onPause -> A:onStop -> A:onDestroy。**

**此时如果按下Home键（非长按），系统返回到桌面，并依次执行A:onPause -> A:onStop。由此可见，Back键和Home键主要区别在于是否会执行onDestroy。**

**此时如果长按Home键，不同手机可能弹出不同内容，Activity生命周期未发生变化（由小米2s测的，不知道其他手机是否会对Activity生命周期有影响）。**

4 . ****启动模式****   
Activity的启动模式决定了激活Activity时，是否创建新的对象，进而将影响到任务栈也叫回退栈。   
在AndroidManifest.xml文件中，可以为每个activity节点配置android:launchMode属性，以决定该Activity的启动模式，该属性的值有：   
—–standard：(默认值)标准模式：每次激活Activity时，都会创建新的Activity对象   
—–singleTop：栈顶时唯一，即当Activity处于栈顶位置时，每次激活并不会创建新的Activity对象。但不在栈顶时，每次激活时会创建新的对象。   
—–singleTask：任务栈中唯一，即当栈中没有该Activity时，将创建该Activity对象，当栈中已经有该Activity时，将不会创建新的对象，原本栈中位于该Activity之上的其它Activity将全部被强制出栈，且被激活的Activity将自动获得栈顶位置。   
—–singleInstance：实例(对象)唯一，确保该Activity的对象一定只有1个，被设置为singleInstance的Activity将被置于一个专门的任务栈中，且该任务栈中有且仅有一个Activity。   
****什么是任务栈（回退栈）：****   
任务栈是用来存放所有激活了的Activity对象，激活的Acitvity将会按照后进先出的栈结构显示出来。因为屏幕只能显示一个Activity，当有新的Activity被激活时，原来正在显示的Activity就会进行压栈操作被压到新Activity对象下方的位置。当按下”Back”键时栈顶Activity会执行弹栈操作，而在第2位的Activity将获得栈顶位置，显示在前台



## 2、**service**

1. ****定义与作用****   
   — Service（服务）是一个没有用户界面的在后台运行执行耗时操作的应用组件。其他应用组件能够启动Service，并且当用户切换到另外的应用场景，Service将持续在后台运行。另外，一个组件能够绑定到一个service与之交互（IPC机制），例如，一个service可能会处理网络操作，播放音乐，操作文件I/O或者与内容提供者（content provider）交互，所有这些活动都是在后台进行，以上是Google文档的解释，资料来源于[大神博客](http://blog.csdn.net/ryantang03/article/details/7770939" \t "https://blog.csdn.net/pigdreams/article/details/_blank)   
   — Service还有一个作用就是提升进程（每个应用都是一个进程）的优先级，进程的优先级指的是在Android系统中，会把正在运行的应用确定一个优先级，当内存空间不足时，系统会根据进程的优先级清理掉一部分进程占用的内存空间，以获得足够的内存空间以供新启用的应用运行。详细的进程优先级划分如下，   
   1)前台进程：应用程序存在Activity正位于前台，可见并可控   
   2)可见进程：应用程序存在Activity处于局部可见状态，即局部可见却不可控   
   3)服务进程：应用程序存在正在运行的Service   
   4)后台进程：应用程序的所有Activity均被置于后台，没有任何Activity可见   
   5) 空进程：已经退出的应用程序   
   service的进程优先级详细介绍请参考这篇博文，[点此进入](http://blog.csdn.net/fhy_2008/article/details/7328967" \o "进程优先级详细介绍" \t "https://blog.csdn.net/pigdreams/article/details/_blank)

2 . ****状态****   
1)启动   
【启动service】   
用Context类定义的startService(Intent)即可启动Service组件，其中intent定义方法与跳转Activity类似，只需把Actvity类换成Service类即可。其生命周期为启动时onCreate()–>onStartCommand()–>销毁时onDestroy(), 反复调用startService()只会导致Service反复执行onStartCommand()   
【停止service】   
调用Context类定义的stopService(Intent)即可停止Service组件，反复调用并没有任何效果，亦不会报告错误，即：即使停止没有启动的Service也不会出错。也可以在Service类的内部，调用Service定义的stopSelf()方法，停止当前Service。

2)绑定   
主要作用是实现组件间的通信，实质的表现是Activity可以调用Service中的方法，使Service执行特定的业务，并且这些方法可以是带返回值的方法，进而Activity可以通过获取这些返回值，这样就实现与Service的通信。   
【生命周期】   
– onCreate() -> 当第1次绑定时执行   
– onBind() -> 当第1次绑定时执行   
– onDestroy() -> 当解除绑定时执行

【绑定与解绑】   
调用bindService()方法可以实现Activity与Service的绑定，调用unbindService()可以解除绑定。在Activity被销毁之前，必须解除与Service的绑定。

## **ContentProvider 了解一下**

****1.作用**** 中文意思是内容提供者，ContentProvider可以将应用程序自身的数据对外(对其它应用程序)共享，使得其它应用可以对自身的数据进行增、删、改、查操作。   
Android系统使用了许多ContentProvider，将系统中的绝大部分常规数据进行对外共享，例如：联系人资料、通话记录、短信、相册、歌曲、视频、日历等等，一般这些数据都存放于一个个的数据库中。

主要用于俩应用数据交互

## **BraodCast Receiver**

**Android的 机制有很多功能都会发送广播的 如 电量变化 锁屏 开屏 开机 关机 等等也可以自己发送**

****1.概述****   
广播接收器，顾名思义这是用于接收应用发送的广播的系统组件。广播是一种1对多的通信方式，即存在1个发送方，若干个接收方。在Android系统，把具有这样的数据的传递方式的机制称之为“广播”。Android系统会在特定的情景下发出各种广播，例如开机、锁屏了、电量不足了、正在充电了、呼出电话了、被呼叫了……   
广播是一种跨进程的、“全设备之内”的通信方式。

****2.发送广播****   
调用Context对象的sendBroadcast(Intent)即可发送广播，在参数Intent对象中，应该调用setAction()方法配置广播的“频道号”，即是相当于收音机要接收某个电台的频段，只有注册了相同的Action的广播接收者才可以接收到该广播。

****3.接收广播****   
自定义类，继承自android.content.BroadcastReceiver后需要注册，注册时，使用IntentFilter配置与发送方相同的Action，重写onReceiver()方法实现对广播的处理。

public class MyBroadcastReceiver extends BroadcastReceiver {

private static final String TAG = "MyBroadcastReceiver";

@Override

public void onReceive(Context context, Intent intent) {

StringBuilder sb = new StringBuilder();

sb.append("Action: " + intent.getAction() + "\n");

sb.append("URI: " + intent.toUri(Intent.URI\_INTENT\_SCHEME).toString() + "\n");

String log = sb.toString();

Log.d(TAG, log);

Toast.makeText(context, log, Toast.LENGTH\_LONG).show();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12

****4.注册广播接收器****   
广播接收者的注册可以区分为静态注册和动态注册：

* 静态注册：在清单文件AndroidManifest.xml中，在application节点下使用receiver节点进行注册。这种方式注册的广播接收者必须是一个单独的只实现BroadcastReceiver的类，不能是内部类。且这样的广播接收者是常驻型的，即从APP安装到手机上的那一刻即开始处于接收广播状态，且直至该APP被从手机移除。

<receiver android:name=".MyBroadcastReceiver" android:exported="true">

<intent-filter>

<action android:name="android.intent.action.BOOT\_COMPLETED"/>

<action android:name="android.intent.action.INPUT\_METHOD\_CHANGED" />

</intent-filter></receiver>

* 1
* 2
* 3
* 4
* 5
* 6
* 动态注册：在程序中调用Context对象的registerReceiver(BroadcastReceiver, IntentFilter)方法进行注册。这种方式可以注册以内部类的形式存在的广播接收者，且这种方式的广播接收者仅当注册后才开始接收广播，并且在调用了Context对象的unregisterReceiver(BroadcastReceiver)方法后就会停止接收广播。

BroadcastReceiver br = new MyBroadcastReceiver();

IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY\_ACTION);

intentFilter.addAction(Intent.ACTION\_AIRPLANE\_MODE\_CHANGED);

this.registerReceiver(br, filter);

* 1
* 2
* 3
* 4

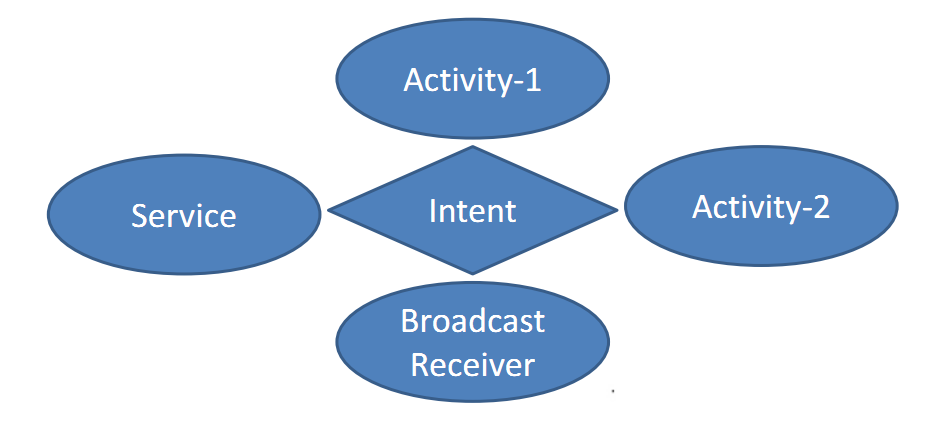
****5.无序广播与有序广播****   
1. 普通的广播即为无序广播，谁都可以接收，并不会相互打扰。   
2. 有序广播：调用sendOrderedBroadcast(Intent, String permission)方法发送的广播，各广播接收者在接收广播时，会存在一定的先后顺序，即某接收者会先收到广播，其他接收者后收到广播，广播会在各接收者之间按照一定的先后顺序进行传递。在广播的传递过程中，先接收到广播的接收者可以对广播进行拦截或篡改。

****6.有序广播的接收者们的优先级****   
有序广播的接收者们的优先级用于确定接收的先后顺序，优先级越高的接收者，将更优先接收到广播，反之，则更靠后接收到广播。   
1. 注册广播时，在广播对应的IntentFilter中的priority属性直接决定优先级，该属性值为int类型的数值，取值越大，则优先级越高！   
2. 如果存在多个广播接收者配置的priority属性值相同，则动态注册的广播接收者的优先级高于静态注册的广播接收者。   
3. 如果根据以上2条规则都无法确定优先级，则根据注册的先后顺序确定各接收者们的优先级。

****7.有序广播的拦截或篡改****   
1. 【拦截】在广播接收者中，使用abortBroadcast()方法，可以终止有序广播向后继续传递，即后续的接收者们将无法接收到该广播。注意：该方法只能在接收有序广播时调用！   
2. 【篡改】在广播接收者中，调用setResult()方法，可以向广播中添加数据，并在后续的接收者中，可以通过getResult()获取这些数据，同时，后续的接收者也可以再次调用setResult()方法重新向广播中写入数据，即覆盖原有的数据，以实现篡改。

Intent 重要程度不下与四大组件

四大组件的联系全靠Intent



Android平台进行数据存储的五大方式,分别如下:

**[1 使用SharedPreferences存储数据](https://www.cnblogs.com/ITtangtang/p/3920916.html" \l "type1) （卸载或者清除缓存丢失）**

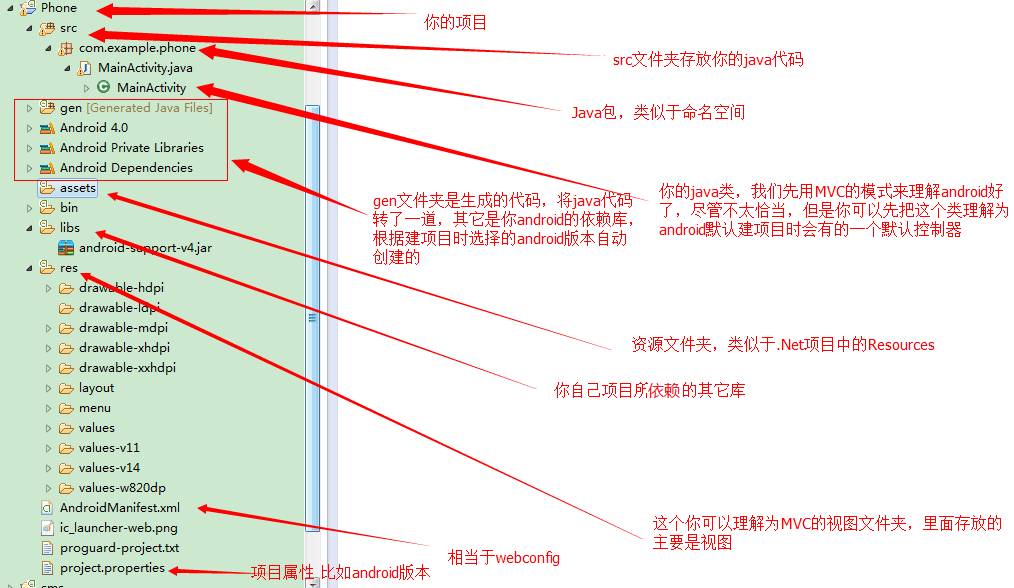
[2 文件存储数据](https://www.cnblogs.com/ITtangtang/p/3920916.html" \l "type2)**（文件不删除永久存在）**

**[3 SQLite数据库存储数据](https://www.cnblogs.com/ITtangtang/p/3920916.html" \l "type3) （1，内部存储与SharedPreferences相同卸载或者清除缓存丢失2、数据库放在文件中）**

**[4 使用ContentProvider存储数据](https://www.cnblogs.com/ITtangtang/p/3920916.html" \l "type4) （等同于2和3）**

**[5 网络存储数据](https://www.cnblogs.com/ITtangtang/p/3920916.html" \l "type5) （类似2和3 但是位置是在服务器 ）**

1、下图为eclipse项目结构 与Android Studio基本相同（不同处不影响理解）



1. 如何去读一个项目
   1. 打开 androidMainifest

首先看**android:name=".App" 如图中 package="com.test.numkeyboardtest"**

**(点代表包名 如果此类不在包名下写App的包名.App(总之.可以代替包名))**

可以没有 此类继承Application

此类可以做一些初始化的操作保证项目启动后 必要数据提前加载好

2、<**activity android:name=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
</**activity**>

其中**.MainActivity为入口类 打开一个项目首先看到的就是他** 此类继承Activity（在这里声明的作用）

<**activity android:name=".MainActivity"**>、、告诉系统我要用那个类

<**intent-filter**> //告诉系统这个来是做入口用的   
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"(告诉系统项目启动先启动我)** />  
</**intent-filter**>

图中的其他 activity标签则由开发人员启动

## 这是**application**标签的作用 其中四大组件都需要在这里声明(**BraodCastReceiver在这里声明可以在项目不启动的情况下接收广播(Android5.0后禁止(加了权限相当于禁止)) 可以在代码中注册(声明)**)

使用ide(开发工具)创建时会自动声明 如复制来的则需要手写