

# 2021

## 第4章 Android生命周期

复旦大学 陈辰



# 学习目标

AIMS

01

了解Android系统的四大基本组件

02

了解Android系统的进程优先级的变化方式

03

了解Activity的生命周期中各状态的变化关系

04

掌握Activity事件回调函数的作用和调用顺序

05

掌握Android应用程序的调试方法和工具

# 4.1

## Android组件

- **Android系统四大组件（调用的基本模块）**

- Activity

- Android程序的呈现层，显示可视化的用户界面，并接收与用户交互所产生的界面事件
    - Android应用程序可以包含一个或多个Activity，一般需要指定一个程序启动时显示的Activity

- Service

- Service一般用于没有用户界面，但需要长时间在后台运行的应用
    - 可公开Service的程序接口，供其他进程调用

# 4.1

## Android组件

- **Android系统四大组件（调用的基本模块）**

- BroadcastReceiver

- 用来接收广播消息的组件，不包含任何用户界面
    - 可以启动Activity或者Notification通知用户接收到重要信息
      - Notification能够通过多种方法提示用户，包括闪动背景灯、震动设备、发出声音或在状态栏上放置一个图标

- ContentProvider

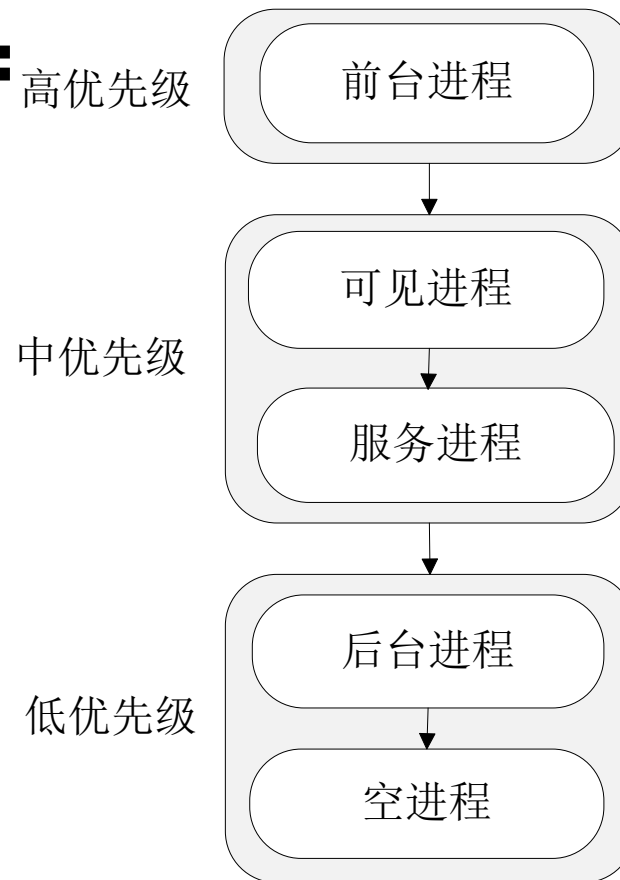
- 是Android系统提供了一种标准的共享数据的机制，其他程序通过ContentProvider访问程序的私有数据
    - Android系统内部提供一些内置的ContentProvider，能够为应用程序提供重要的数据信息
      - 联系人信息
      - 通话记录



## 4.2 程序生命周期

• **Android系统的进程优先级从高到低分别为：**

- 前台进程
- 可见进程
- 服务进程
- 后台进程
- 空进程



## 4.2 程序生命周期

- **前台进程**

- 前台进程是Android系统中最重要的进程
  - 进程中的Activity正在与用户进行交互
  - Service被Activity调用，而且这个Activity正在与用户进行交互
  - Service正在执行声明周期中的回调函数，如onCreate()、onStart()或onDestroy()
  - 进程的BroadcastReceiver正在执行onReceive()函数

## 4.2 程序生命周期

### ■可见进程

- 可见进程指部分程序界面能够被用户看见，却不在前台与用户交互，不响应界面事件的进程
- 如果一个进程包含Service，且这个Service正在被用户可见的Activity调用，此进程同样被视为可见进程

### ■服务进程

- 包含已启动服务的进程
- Android系统除非不能保证前台进程或可视进程所必要的资源，否则不强行清除服务进程

## 4.2 程序生命周期

- **后台进程**

- 指不包含任何已经启动的服务，而且没有任何用户可见的Activity的进程
- Android系统中一般存在数量较多的后台进程

- **空进程**

- 空进程是不包含任何活跃组件的进程



# 4.2

## 程序生命周期

- **优先级决定与变化规则**

- 进程的优先级取决于所有组件中的优先级最高的部分
- 进程的优先级会根据与其他进程的依赖关系而变化

# 4.3

## Android组件

- **组件生命周期**

- 所有Android组件都具有自己的生命周期，是从组件建立到组件销毁的整个过程
- 在生命周期中，组件会在可见、不可见、活动、非活动等状态中不断变化

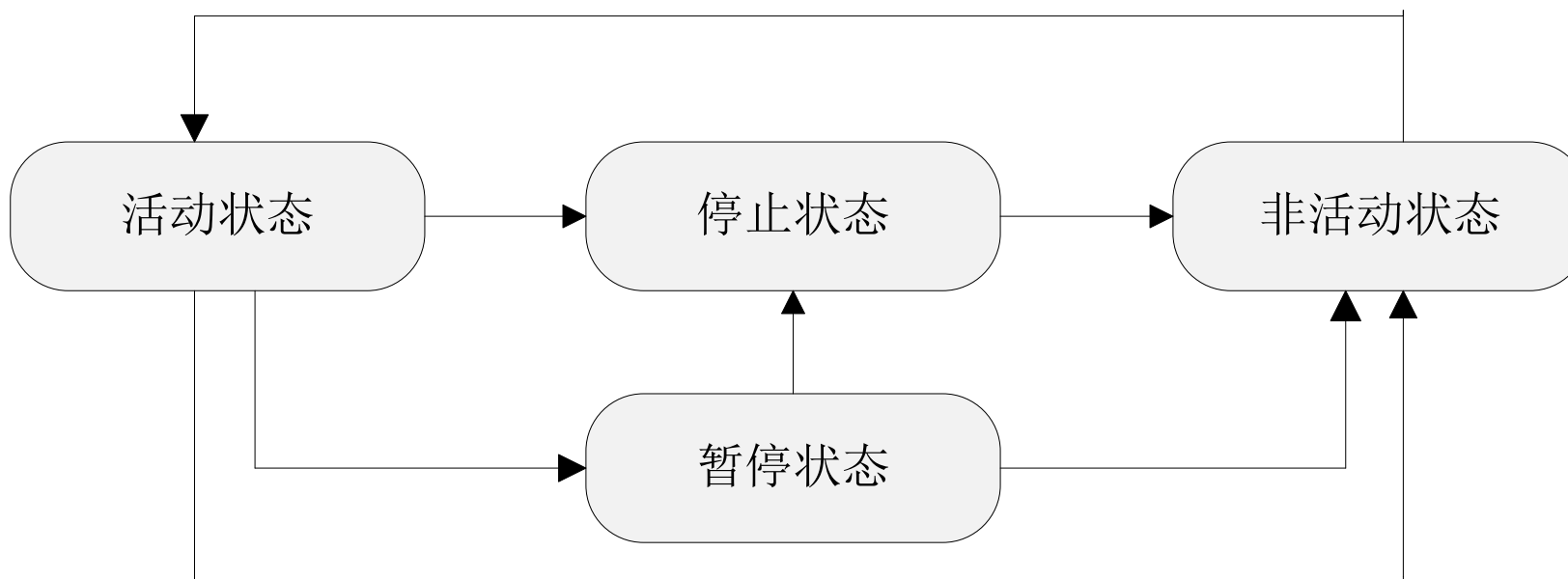
## 4.3 Activity生命周期

- **Activity生命周期**

- Activity生命周期指Activity从启动到销毁的过程
- Activity表现为四种状态
  - 活动状态，Activity在用户界面中处于最上层，完全能不用户看到，能够与用户进行交互
  - 暂停状态，Activity在界面上被部分遮挡，该Activity不再处于用户界面的最上层，且不能够与用户进行交互
  - 停止状态，Activity在界面上完全不能被用户看到，也就是说这个Activity被其他Activity全部遮挡
  - 非活动状态，不在以上三种状态中的Activity则处于非活动状态

## 4.3 Activity生命周期

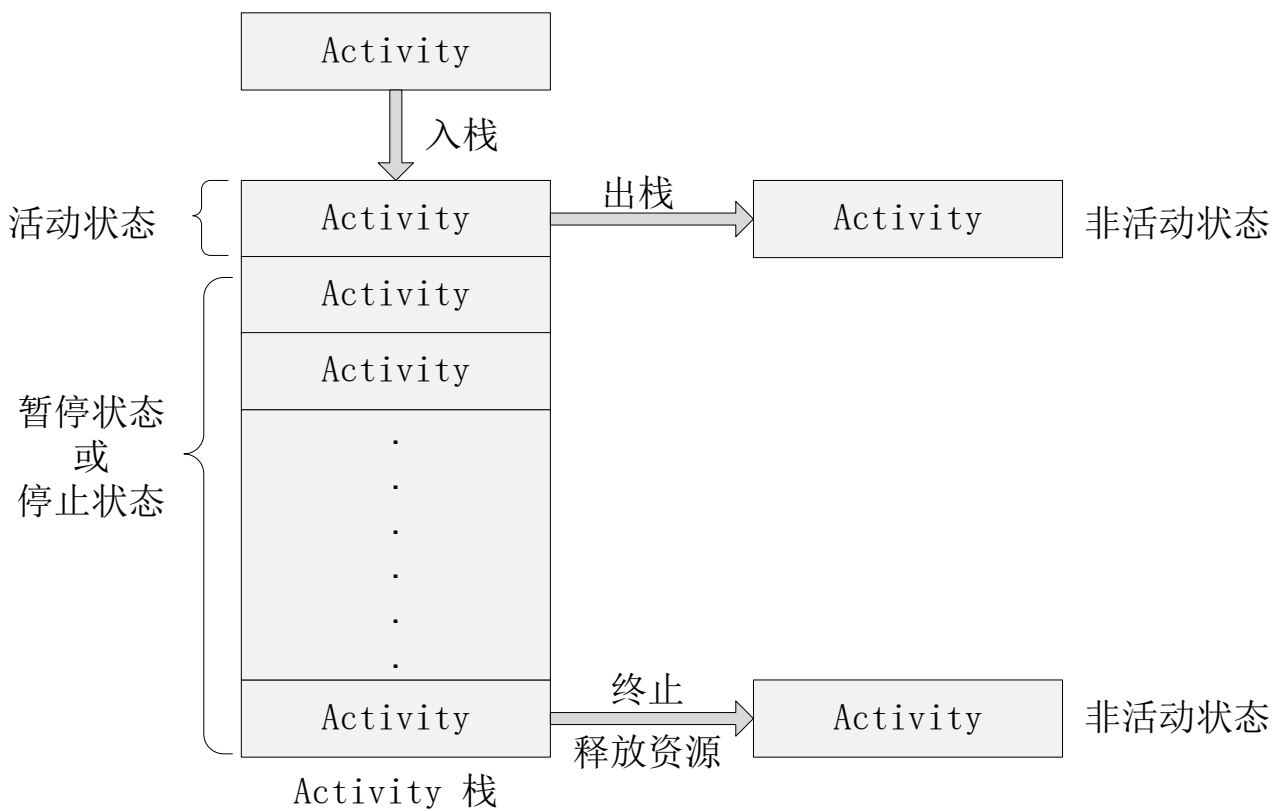
### • Activity的四种状态的变换关系图



## 4.3 Activity生命周期

- **Activity栈**

- 遵循“后进先出”的规则





## 4.3 Activity生命周期

- 随着Activity自身状态的变化，Android系统会调用不同的事件回调函数
  - Activity的主要事件回调函数

```
1. public class MyActivity extends Activity {  
2.     protected void onCreate(Bundle savedInstanceState);  
3.     protected void onStart();  
4.     protected void onRestart();  
5.     protected void onResume();  
6.     protected void onPause();  
7.     protected void onStop();  
8.     protected void onDestroy();  
9. }
```

## 4.3 Activity生命周期

- 函数

- Activity生命周期的事件回调函数

函数	是否可终止	说明
onCreate()	否	Activity启动后第一个被调用的函数，常用来进行Activity的初始化，例如创建View、绑定数据或恢复信息等。
onStart()	否	当Activity显示在屏幕上时，该函数被调用。
onRestart()	否	当Activity从停止状态进入活动状态前，调用该函数。
onResume()	否	当Activity可以接受用户输入时，该函数被调用。
onPause()	否	当Activity进入暂停状态时，该函数被调用。主要用来保存持久数据、关闭动画、释放CPU资源等。该函数中的代码必须简短，因为另一个Activity必须等待该函数执行完毕后才能显示在界面上。
onStop()	是	当Activity不对用户可见后，该函数被调用，Activity进入停止状态。
onDestroy()	是	在Activity被终止前，即进入非活动状态前，该函数被调用。有两种情况该函数会被调用：(1)当程序主动调用finish()函数；(2)程序被Android系统终结。

## 4.3 Activity生命周期

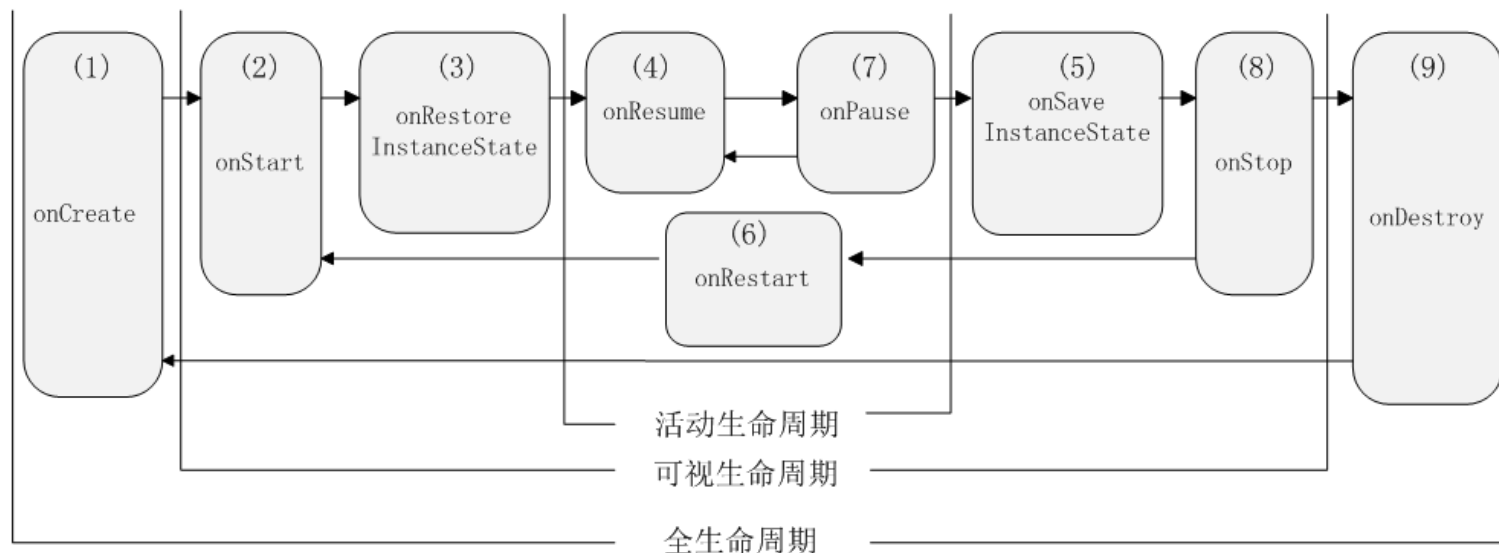
- **onRestoreInstanceState()和onSaveInstanceState()**

- 不属于生命周期的事件回调函数，但可以用于保存和恢复Activity的界面临时信息
- onSaveInstanceState()会将界面的临时信息保存在Bundle中， onCreate()和 onRestoreInstanceState()都可以恢复这些保存的信息
- 简化的办法是使用onCreate() 恢复， 但有些特殊的情况下还是需要使用 onRestoreInstanceState()函数恢复
  - 必须在界面完全初始化完毕后才能进行的操作
  - 或需要由子类来确定是否采用缺省设置等

函数	说明
onSaveInstanceState()	暂停或停止Activity前调用该函数，用以保存Activity的临时状态信息
onRestoreInstanceState()	恢复onSaveInstanceState()保存的Activity状态信息。

## 4.3 Activity生命周期

### • Activity事件回调函数的调用顺序



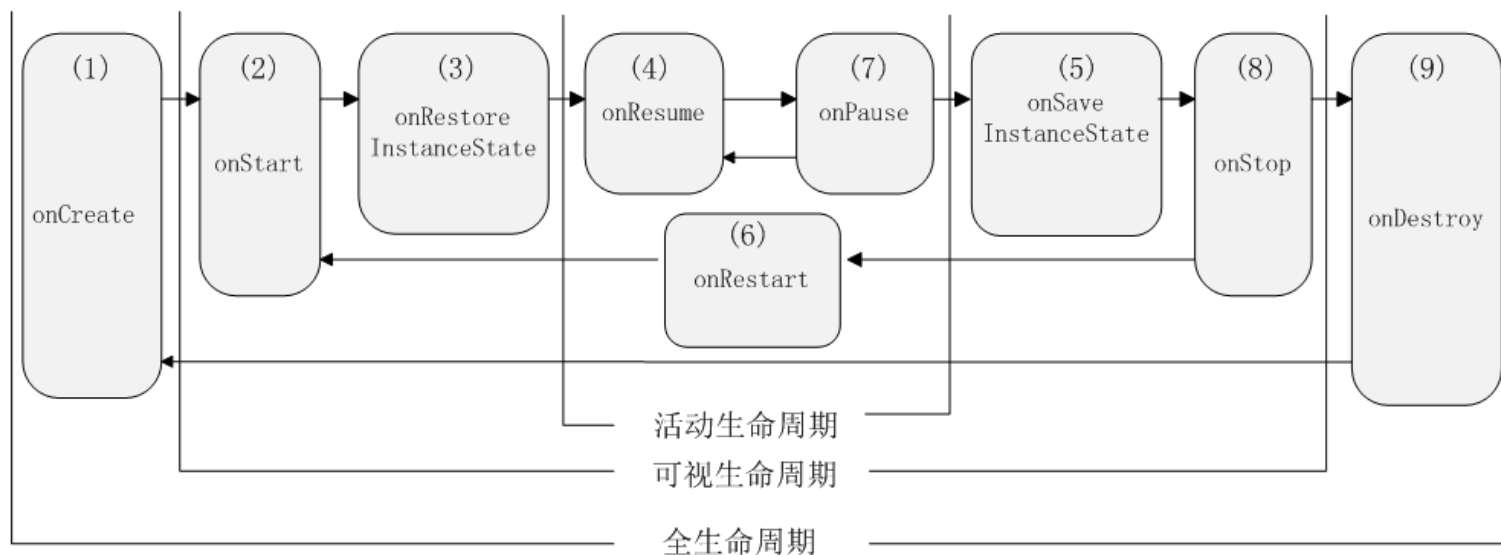
### • Activity生命周期分类

- Activity的生命周期可分为全生命周期、可视生命周期和活动生命周期
- 每种生命周期中包含不同的事件回调函数

## 4.3 Activity生命周期

### • 全生命周期

- 全生命周期是从Activity建立到销毁的全部过程，始于onCreate()，结束于onDestroy()
- 使用者通常在onCreate()中初始化Activity所能使用的全局资源和状态，并在onDestroy()中释放这些资源
- 在一些极端的情况下，Android系统会不调用onDestroy()函数，而直接终止进程

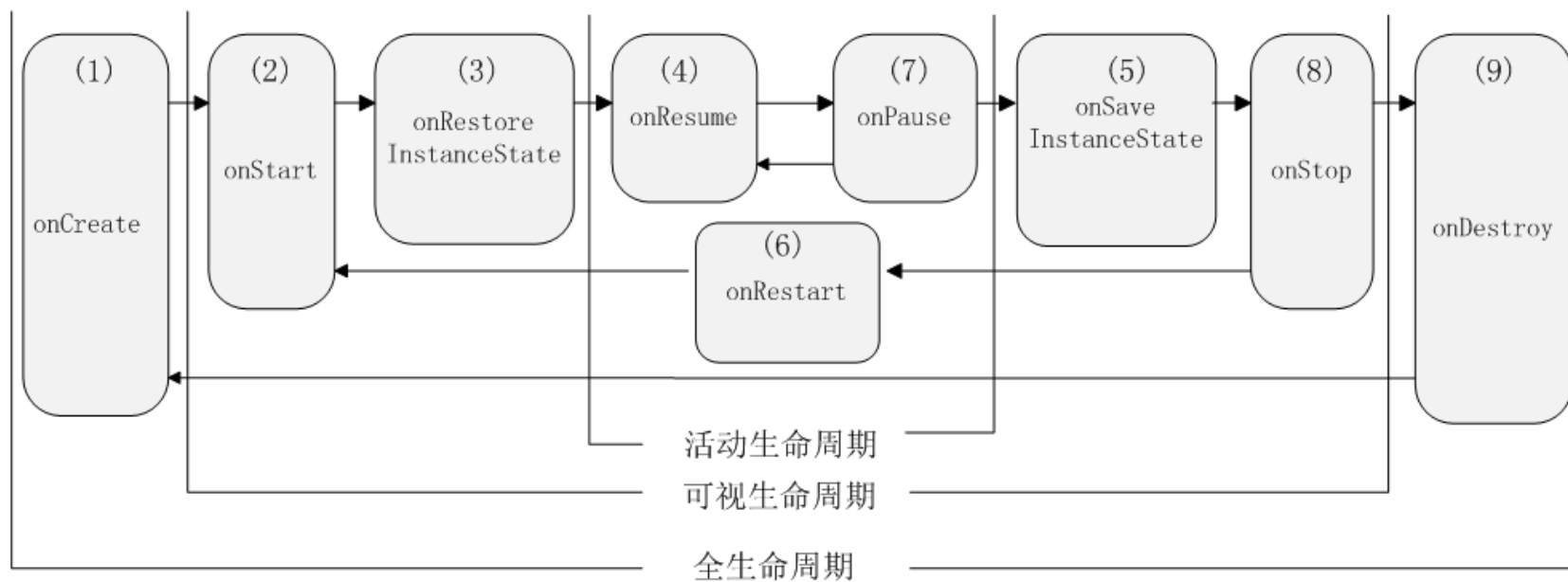




## 4.3 Activity生命周期

- 可视生命周期

- 可视生命周期是Activity在界面上从可见到不可见的过程，开始于onStart()，结束于onStop()



## 4.3 Activity生命周期

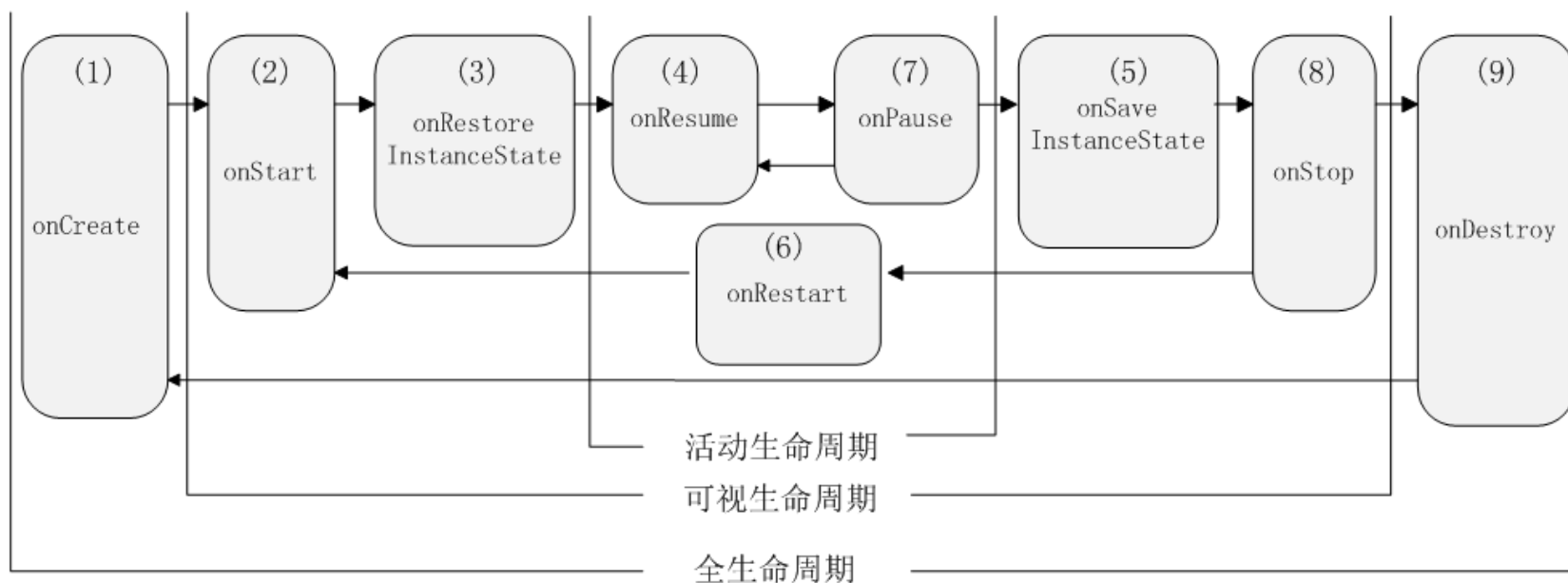
- 可视生命周期

- onStart()
  - 一般用来初始化或启动与更新界面相关的资源
- onStop()
  - 一般用来暂停或停止一切与更新用户界面相关的线程、计时器和服务
- onRestart()
  - 函数在onStart()前被调用，用来在Activity从不可见变为可见的过程中，进行一些特定的处理过程
- onStart()和onStop()会被多次调用

## 4.3 Activity生命周期

### • 活动生命周期

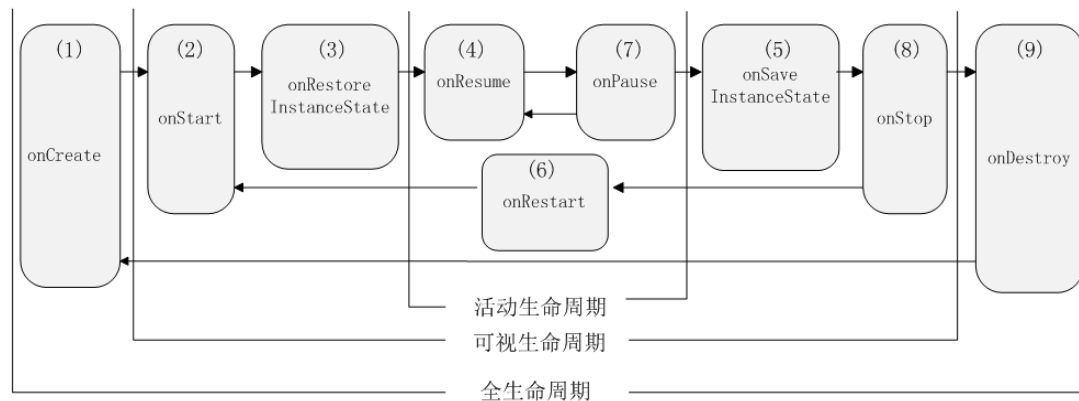
- 活动生命周期是Activity在屏幕的最上层，并能够与用户交互的阶段，开始于onResume()，结束于onPause()
- 在Activity的状态变换过程中onResume()和onPause()经常被调用，因此这两个函数中应使用更为简单、高效的代码



## 4.3 Activity生命周期

- **onPause()和onSaveInstanceState()**

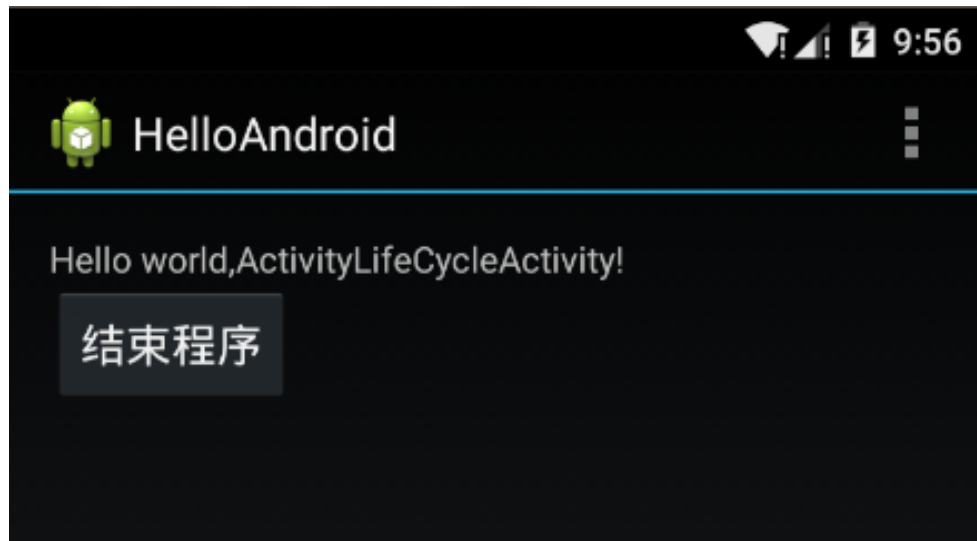
- 这两个函数都可以用来保存界面的用户输入数据
- onPause()一般用于保存持久性数据，并将数据保存在存储设备上的文件系统或数据库系统中的
- onSaveInstanceState()主要用来保存动态的状态信息，信息一般保存在Bundle中
  - Bundle是能够保存多种格式数据的对象，系统在调用onRestoreInstanceState()和onCreate()时 会同样利用Bundle将数据传递给函数



## 4.3 Activity生命周期

- 活动生命周期

- 下面以ActivityLifeCycle示例来进行说明， ActivityLifeCycle示例的运行界面





## 4.3 Activity生命周期

### • ActivityLifecycleActivity.java文件的全部代码

```
1. package edu.hrbeu.ActivityLifeCycle;
2.
3. import android.app.Activity;
4. import android.os.Bundle;
5. import android.util.Log;
6. import android.view.View;
7. import android.widget.Button;
8.
9. public class ActivityLifeCycle extends Activity {
10.     private static String TAG = "LIFTCYCLE";
11.     @Override //完全生命周期开始时被调用，初始化Activity
12.     public void onCreate(Bundle savedInstanceState) {
13.         super.onCreate(savedInstanceState);
14.         setContentView(R.layout.main);
15.         Log.i(TAG, "(1) onCreate()");
16.     }
17.
```

# 4.3

## Activity生命周期

```
18.  @Override //可视生命周期开始时被调用，对用户界面进行必要的更改
19.      public void onStart() {
20.          super.onStart();
21.          Log.i(TAG, "(2) onStart()");
22.      }
23.
24.      @Override //在onStart()后被调用，用于恢复onSaveInstanceState()保存的用户界面信息
25.      public void onRestoreInstanceState(Bundle savedInstanceState) {
26.          super.onRestoreInstanceState(savedInstanceState);
27.          Log.i(TAG, "(3) onRestoreInstanceState()");
28.      }
29.
30.      @Override //在活动生命周期开始时被调用，恢复被onPause()停止的用于界面更新的资源
31.      public void onResume() {
32.          super.onResume();
33.          Log.i(TAG, "(4) onResume()");
34.      }
35.  33.
36.      @Override // 在onResume()后被调用，保存界面信息
37.      public void onSaveInstanceState(Bundle savedInstanceState) {
```

# 4.3

## Activity生命周期

```
38.     super.onSaveInstanceState(savedInstanceState);
39.         Log.i(TAG, "(5) onSaveInstanceState()");
40.     }
41.
42.     @Override //在重新进入可视生命周期前被调用，载入界面所需要的更改信息
43.     public void onRestart() {
44.         super.onRestart();
45.         Log.i(TAG, "(6) onRestart()");
46.     }
47.
48.     @Override //在活动生命周期结束时被调用，用来保存持久的数据或释放占用的资源
49.     public void onPause() {
50.         super.onPause();
51.         Log.i(TAG, "(7) onPause()");
52.     }
53.
54.     @Override //在可视生命周期结束时被调用，一般用来保存持久的数据或释放占用的资源
```

## 4.3 Activity生命周期

### • ActivityLifecycle.java文件的代码

```
54. public void onStop() {  
55.     super.onStop();  
56.     Log.i(TAG, "(8) onStop()");  
57. }  
58.  
59. @Override //在完全生命周期结束时被调用，释放资源，包括线程、数据连接  
    等  
60. public void onDestroy() {  
61.     super.onDestroy();  
62.     Log.i(TAG, "(9) onDestroy()");  
63. }  
64. }
```

- 上面的程序主要通过生命周期函数中添加“日志点”的方法进行调试，程序的运行结果将会显示在LogCat中
- 为了显示结果易于观察和分析，在LogCat设置过滤器LifecycleFilter，过滤方法选择by Log Tag，过滤关键字为LIFTCYCLE

## 4.3 Activity生命周期

- 全生命周期

- 启动和关闭ActivityLifecycle 的LogCat输出
  - 启动ActivityLifecycle
  - 按下模拟器的“返回键”
  - 关闭ActivityLifecycle
- LogCat输出结果

Level	Time	PID	Application	Tag	Text
I	10-21 01:13:12.947	578	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(1) onCreate()
I	10-21 01:13:12.947	578	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(2) onStart()
I	10-21 01:13:12.947	578	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(4) onResume()
I	10-21 01:17:23.328	578	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(7) onPause()
I	10-21 01:17:25.817	578	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(8) onStop()
I	10-21 01:17:25.817	578	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(9) onDestroy()

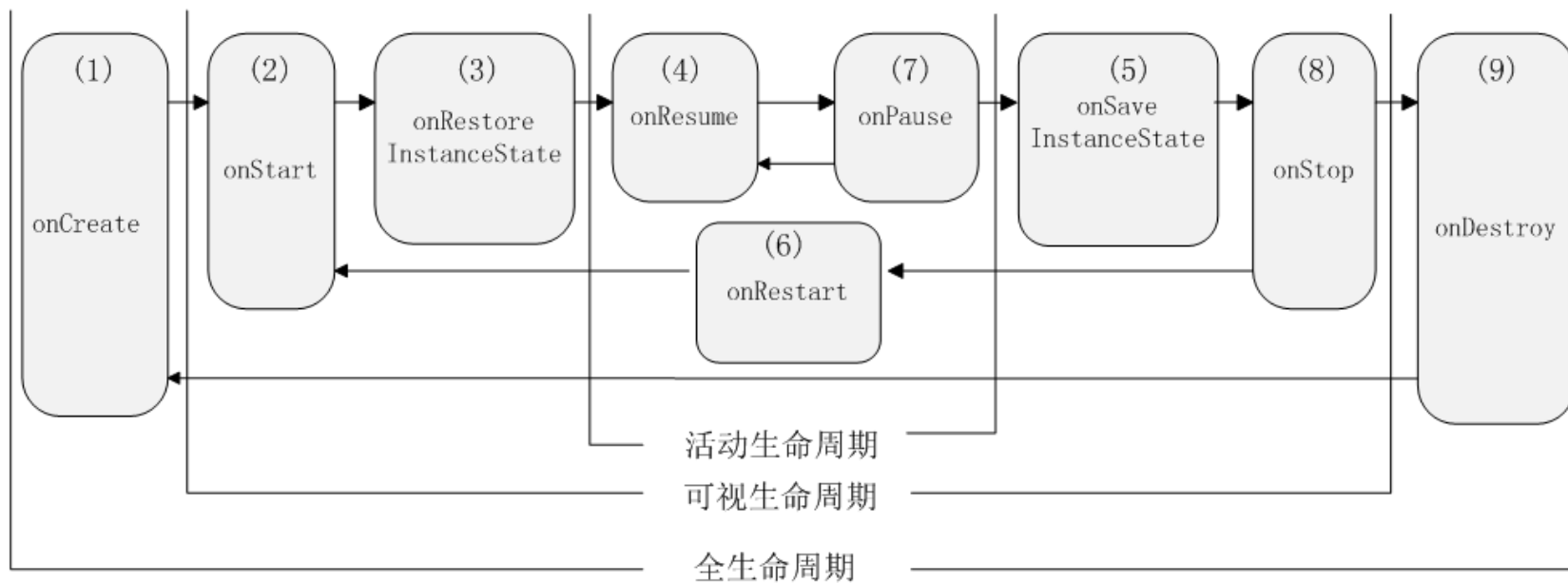


## 4.3 Activity生命周期

### • 全生命周期

- 函数的调用顺序:

- (1)onCreate → (2)onStart → (3)onRestoreInstanceState → (4)onResume →→
- (7)onPause → (8)onStop → (9)onDestroy



# 4.3 Activity生命周期

- 可视生命周期
  - 状态转换
    - 启动ActivityLifecycle
    - 按“呼出/接听键”启动内置的拨号程序
    - 再通过“返回键”退出拨号程序
    - ActivityLifecycle重新显示在屏幕中
  - 可视生命周期的LogCat输出结果

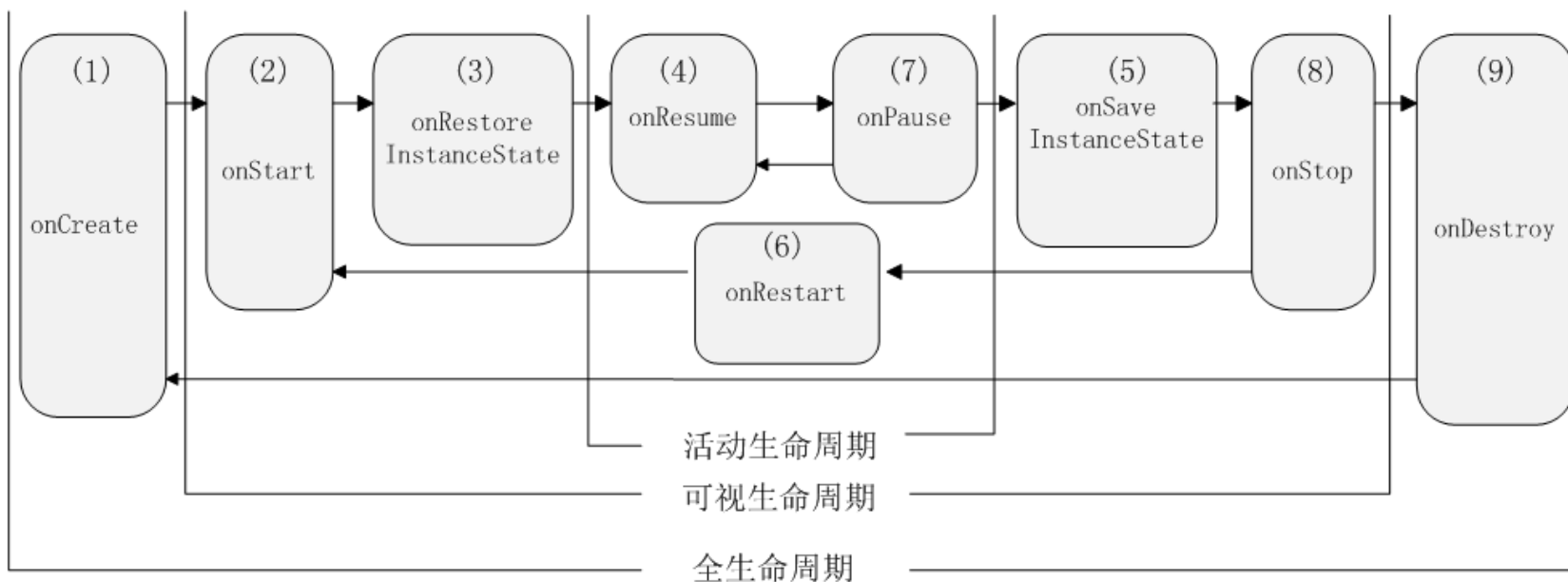
Level	Time	PID	Application	Tag	Text
I	10-22 06:57:24.946	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(1) onCreate ()
I	10-22 06:57:25.056	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(2) onStart ()
I	10-22 06:57:25.056	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(4) onResume ()
I	10-22 06:57:31.687	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(7) onPause ()
I	10-22 06:57:34.276	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(5) onSaveInstanceState ()
I	10-22 06:57:34.329	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(8) onStop ()
I	10-22 06:57:39.867	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(6) onRestart ()
I	10-22 06:57:39.867	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(2) onStart ()
I	10-22 06:57:39.886	567	edu.hrbeu.ActivityLifecycle	LIFECYCLE	(4) onResume ()

## 4.3 Activity生命周期

### • 可视生命周期

- 函数的调用顺序:

- (1)onCreate → (2)onStart → (4)onResume → →
- (7)onPause → (5)onSaveInstanceState → (8)onStop → →
- (6)onRestart → (2)onStart → (4)onResume



## 4.3 Activity生命周期

- 可视生命周期

- 开启IDA的可视生命周期: Dev Tools → Developer options → Don't keep activities 开启该选项

Level	Time	PID	Application	Tag	Text
I	10-22 07:13:33.476	772	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(1) onCreate()
I	10-22 07:13:33.476	772	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(2) onStart()
I	10-22 07:13:33.486	772	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(4) onResume()
I	10-22 07:13:47.946	772	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(7) onPause()
I	10-22 07:13:50.966	772	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(5) onSaveInstanceState()
I	10-22 07:13:50.966	772	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(8) onStop()
I	10-22 07:14:00.287	833	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(1) onCreate()
I	10-22 07:14:00.296	833	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(2) onStart()
I	10-22 07:14:00.327	833	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(3) onRestoreInstanceState()
I	10-22 07:14:00.336	833	edu.hrbeu.ActivityLifeCycle	LIFECYCLE	(4) onResume()

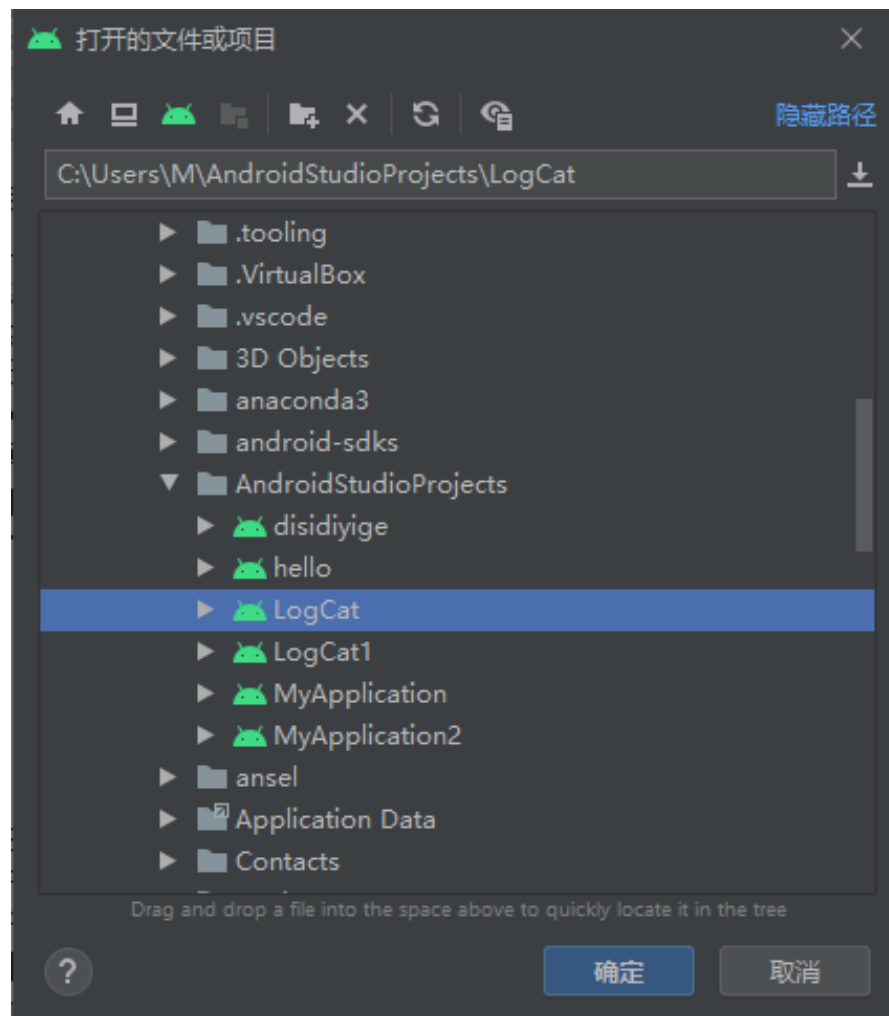
## 4.4 程序调试

- **Android系统提供了两种调试工具LogCat和DevTools，用于定位、分析及修复程序中出现的错误**
- **LogCat**
  - LogCat是用来获取系统日志信息的工具，并可以显示在Eclipse集成开发环境中
  - 功能：能够捕获的信息包括Dalvik虚拟机产生的信息、进程信息、ActivityManager信息、PackageManager信息、HomeLoader 信息、WindowsManager信息、Android运行时信息和应用程序信息等

## 4.4 程序调试

### • 4.4.1 LogCat

- 在Android Studio的开发模式下，用户可以使用文件→打开→打开的文件或项目，然后在AndroidStudioProjects→ LogCat中选择LogCat

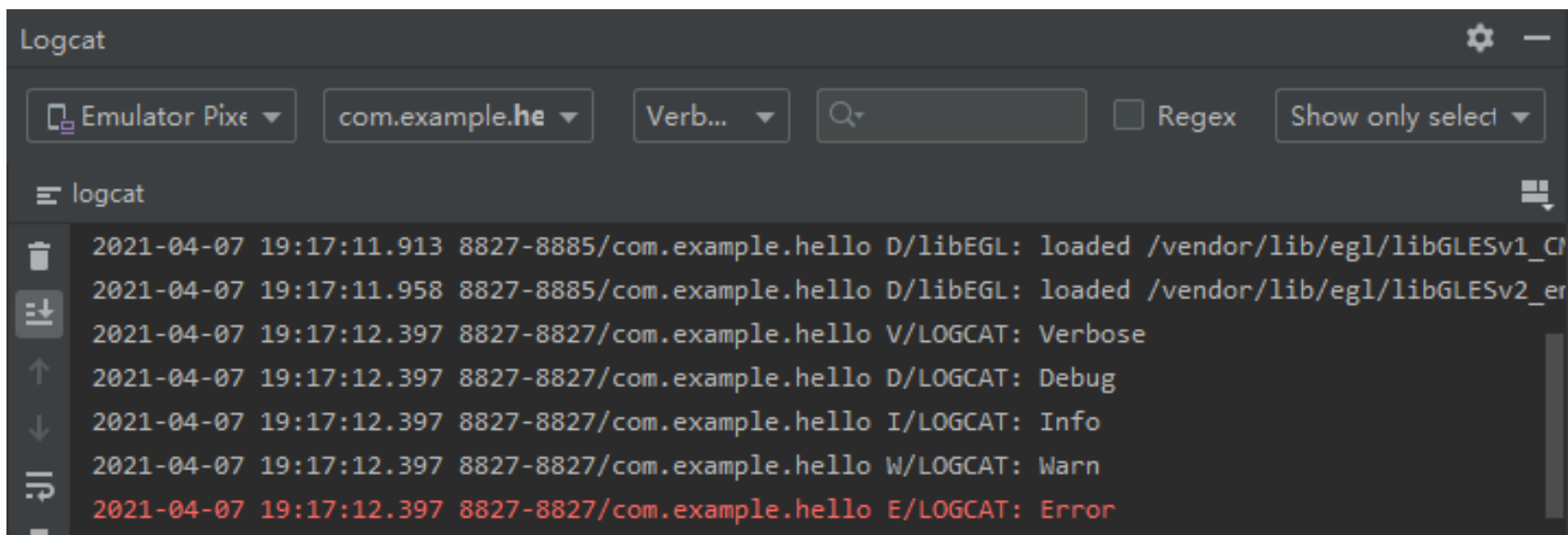




## 4.4 程序调试

### • 4.4.1 LogCat

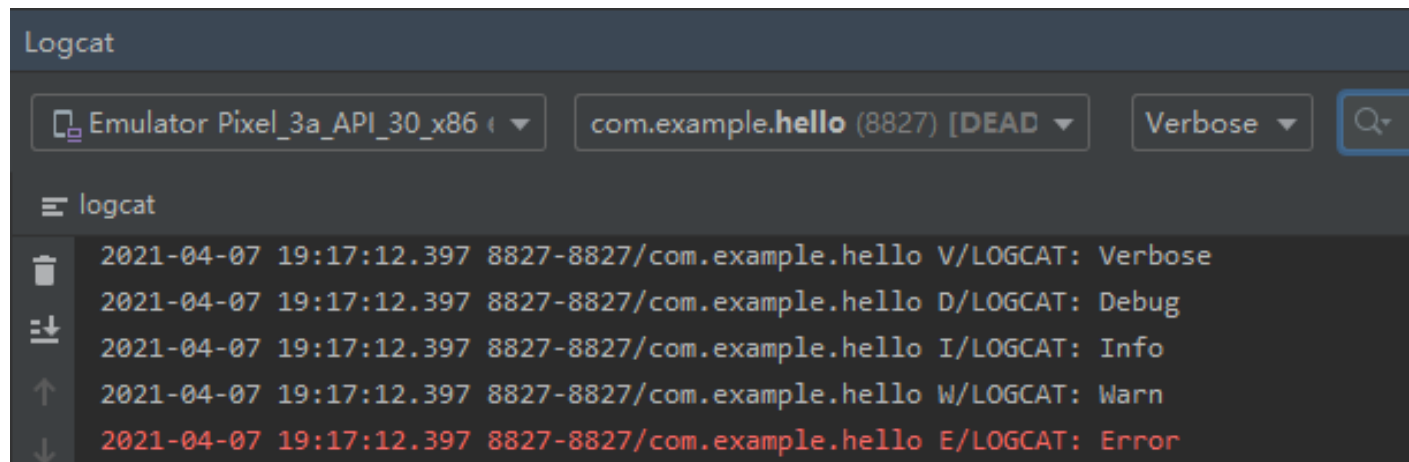
- LogCat便显示在Android Studio的下方区域



# 4.4 程序调试

## • 4.4.1 LogCat

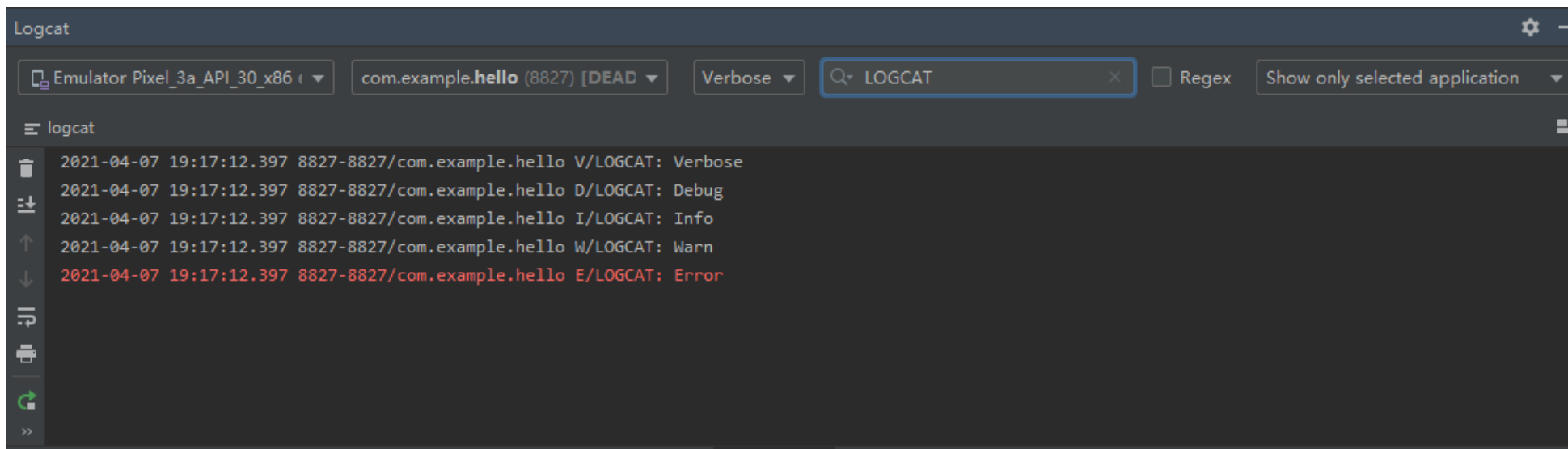
- LogCat的右上方的五个字母表示五种不同类型的日志信息，级别依次增高
- [V]: 详细 (Verbose) 信息
- [D]: 调试 (Debug) 信息
- [I]: 通告 (Info) 信息
- [W]:警告 (Warn) 信息
- [E]:错误 (Error) 信息



# 4.4 程序调试

- **4.4.1 LogCat**

- LogCat提供了“过滤”功能



## 4.4 程序调试

- **4.4.1 LogCat**

- 程序调试原理

- 引入android.util.Log包
    - 使用Log.v()、 Log.d()、 Log.i() 、 Log.w() 和 Log.e()五个函数在程序中设置“日志点”
    - 当程序运行到“日志点”时，应用程序的日志信息便被发送到LogCat中， 判断“日志点”信息与预期的内容是否一致， 进而判断程序是否存在错误

# 4.4 程序调试

## • 4.4.1 LogCat

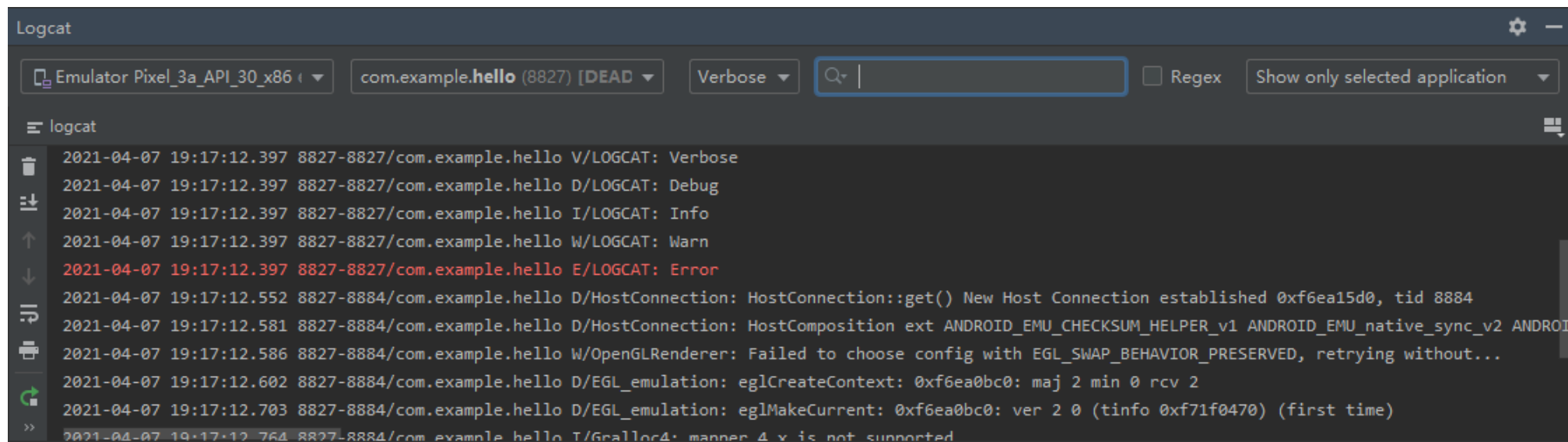
- 演示了Log类的具体使用方法

```
1. package edu.hrbeu.LogCat;
2. import android.app.Activity;
3. import android.os.Bundle;
4. import android.util.Log;
5. public class LogCatActivity extends Activity {
6.     final static String TAG = "LOGCAT";
7.     @Override
8.     public void onCreate(Bundle savedInstanceState) {
9.         super.onCreate(savedInstanceState);
10.        setContentView(R.layout.main);
11.
12.        Log.v(TAG, "Verbose");
13.        Log.d(TAG, "Debug");
14.        Log.i(TAG, "Info");
15.        Log.w(TAG, "Warn");
16.        Log.e(TAG, "Error");
17.    }
18. }
```

# 4.4 程序调试

## • 4.4.1 LogCat

- LogCat工程的运行结果



The screenshot shows the Logcat window in Android Studio. The top bar displays the selected application as 'Emulator Pixel\_3a\_API\_30\_x86' and the package name 'com.example.hello (8827) [DEAD]'. The log level is set to 'Verbose'. A search bar is present with a magnifying glass icon. The log messages are as follows:

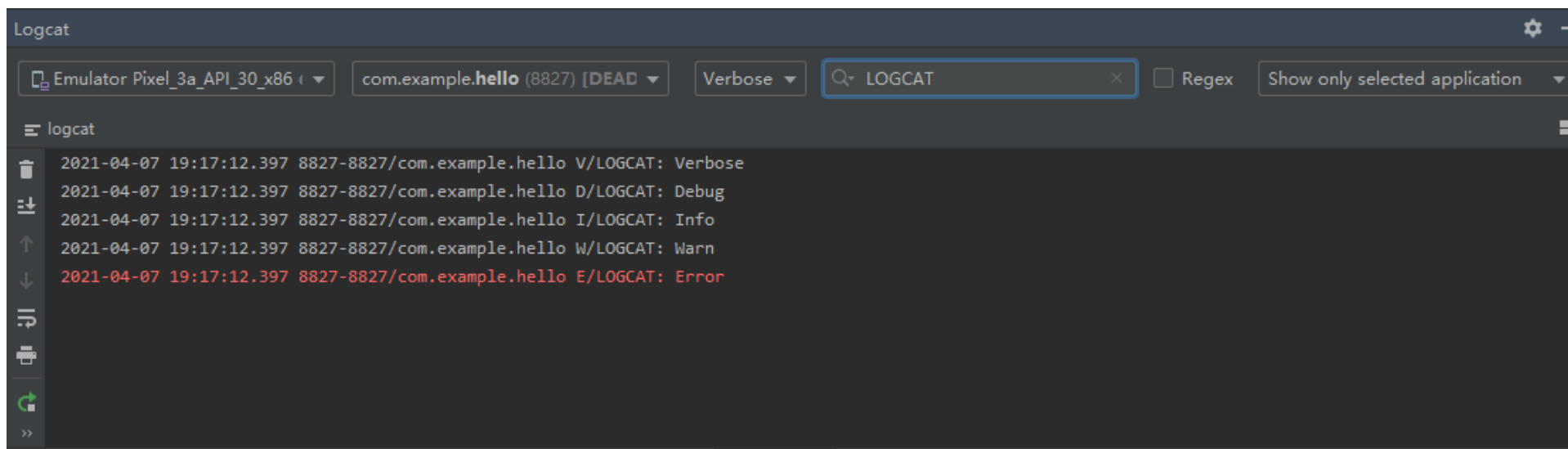
```
logcat
2021-04-07 19:17:12.397 8827-8827/com.example.hello V/LOGCAT: Verbose
2021-04-07 19:17:12.397 8827-8827/com.example.hello D/LOGCAT: Debug
2021-04-07 19:17:12.397 8827-8827/com.example.hello I/LOGCAT: Info
2021-04-07 19:17:12.397 8827-8827/com.example.hello W/LOGCAT: Warn
2021-04-07 19:17:12.397 8827-8827/com.example.hello E/LOGCAT: Error
2021-04-07 19:17:12.552 8827-8884/com.example.hello D/HostConnection: HostConnection::get() New Host Connection established 0xf6ea15d0, tid 8884
2021-04-07 19:17:12.581 8827-8884/com.example.hello D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID
2021-04-07 19:17:12.586 8827-8884/com.example.hello W/OpenGLESRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
2021-04-07 19:17:12.602 8827-8884/com.example.hello D/EGL_emulation: eglCreateContext: 0xf6ea0bc0: maj 2 min 0 rcv 2
2021-04-07 19:17:12.703 8827-8884/com.example.hello D/EGL_emulation: eglMakeCurrent: 0xf6ea0bc0: ver 2 0 (tinfo 0xf71f0470) (first time)
2021-04-07 19:17:12.764 8827-8884/com.example.hello I/Gralloc4: manner 4 x is not supported
```



## 4.4 程序调试

### • 4.4.1 LogCat

- 设置过滤条件为“标签=LOGCAT”， LogCat过滤后的输入结果



- 无论什么类型的日志信息，属于哪一个进程，只要标签为LOGCAT，都将显示在 LogcatFilter区域内

## 4.4 程序调试

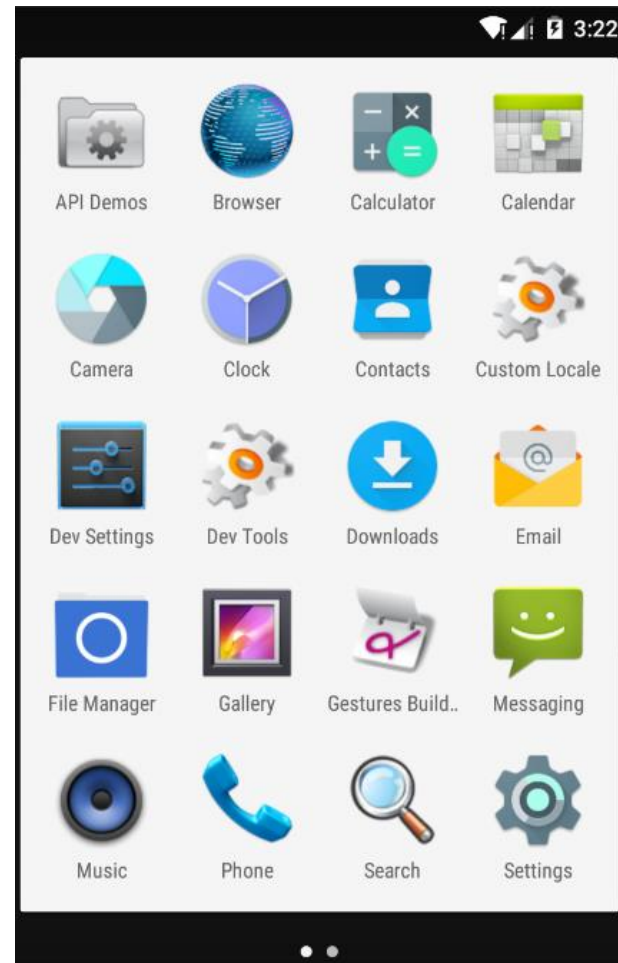
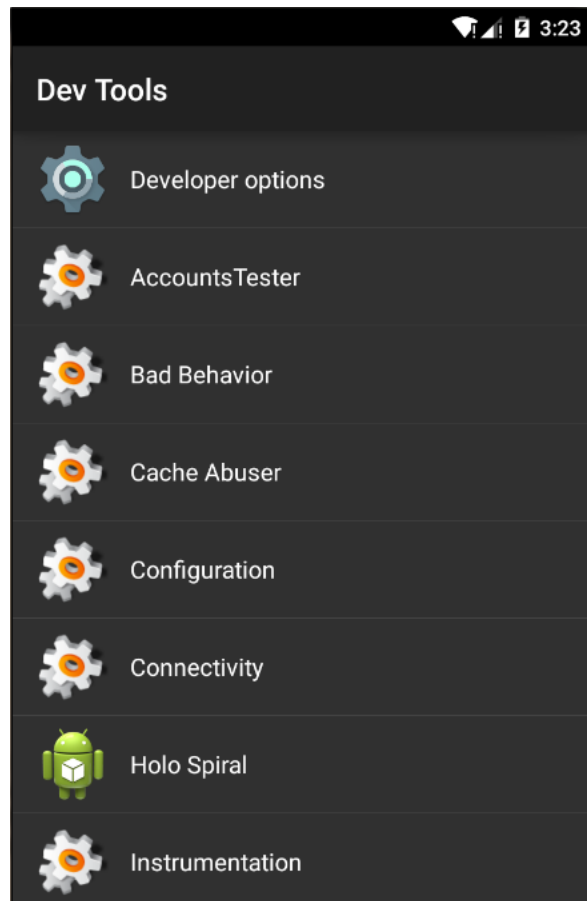
- **4.4.2 DevTools**

- DevTools是用于调试和测试的工具
- 包括了一系列用户各种用途的小工具： Development Settings、Exception Browser、Google Login Service、Instrumentation、Media Scanner、Package Browser、Pointer Location、Raw Image Viewer、Running processes和Terminal Emulator

# 4.4 程序调试

## • 4.4.2 DevTools

- DevTools的使用界面

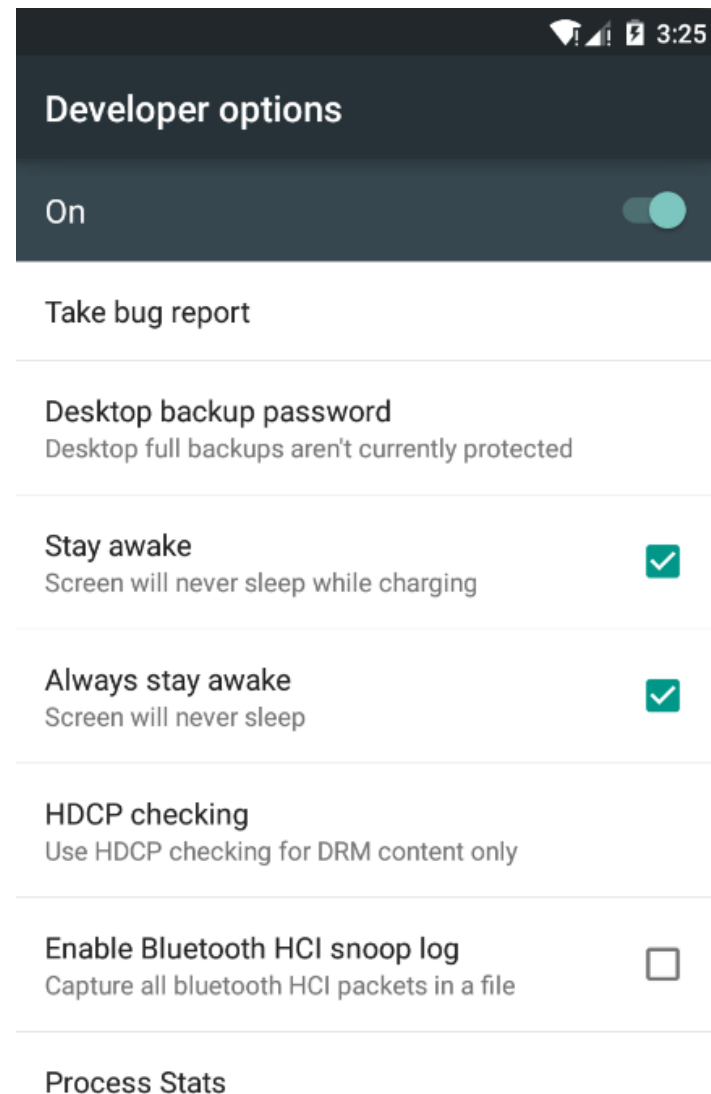


# 4.4 程序调试

## • 4.4.2 DevTools

### • Developer options

- 如果希望启动Developer options中某项功能，只需要点击功能前面选择框出现绿色的“对号”表示功能启用。
- 功能启用后，模拟器会自动保存设置，即使再次启动模拟器用户的选择内容仍会存在



# 4.4 程序调试

- 4.4.2 DevTools

- Developer options选项

选项	说明
Stay awake	当充电时，不锁定屏幕。
Always stay awake	一直不自动锁定屏幕。
USB debugging	不启用该选项就不能使用adb连接设备和开发环境。
Show Touches	显示触摸操作。
Pointer Location	指针位置。
Show CPU usage	在屏幕顶端显示CPU使用率，上层红线显示总的CPU使用率，下层绿线显示当前进程的CPU使用率。
Show background	应用程序没有Activity显示时，直接显示背景面板，一般这种情况仅在调试时出现。
Show sleep state on LED	在休眠状态下开启LED。
Windows Animation Scale	窗口动画模式
Transition Animation Scale	渐变动画模式
Don't keep activities	Activity进入停止状态后立即销毁，用于测试在函数 onSaveInstanceState()、onRestoreInstanceState()和 onCreate()中的代码。
Background Process Limit	限制后台进程的个数

## 4.4 程序调试

- **4.4.2 DevTools**

- Package Browser

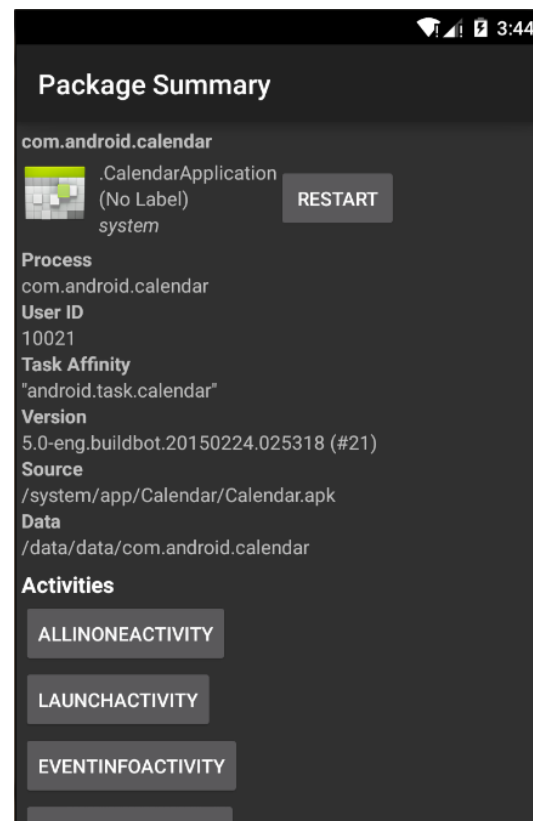
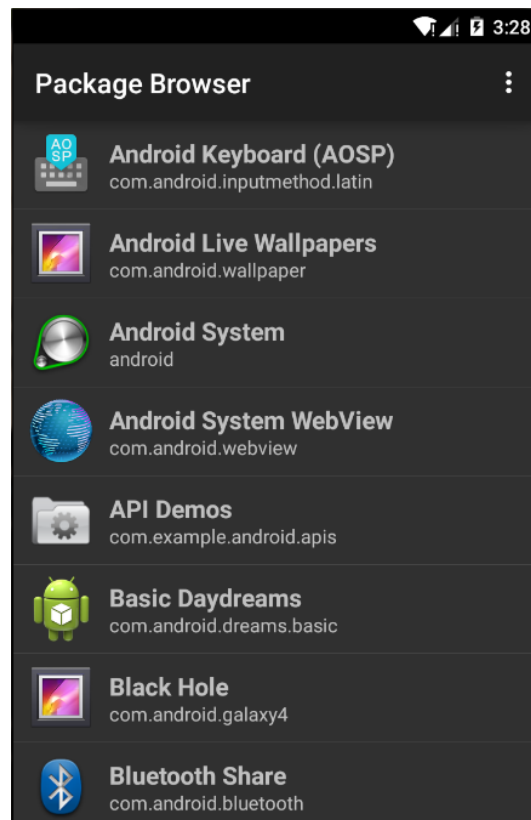
- Package Browser是Android系统中的程序包查看工具，能够详细显示已经安装到Android系统中的程序信息，包括包名称、应用程序名称、图标、进程、用户ID、版本、apk文件保存位置和数据文件保存位置
    - 进一步查看应用程序所包含Activity、Service、BroadcastReceiver和Provider的详细信息



# 4.4 程序调试

## • 4.4.2 DevTools

- 在Package Browser中查看Android keyboard程序的相关信息

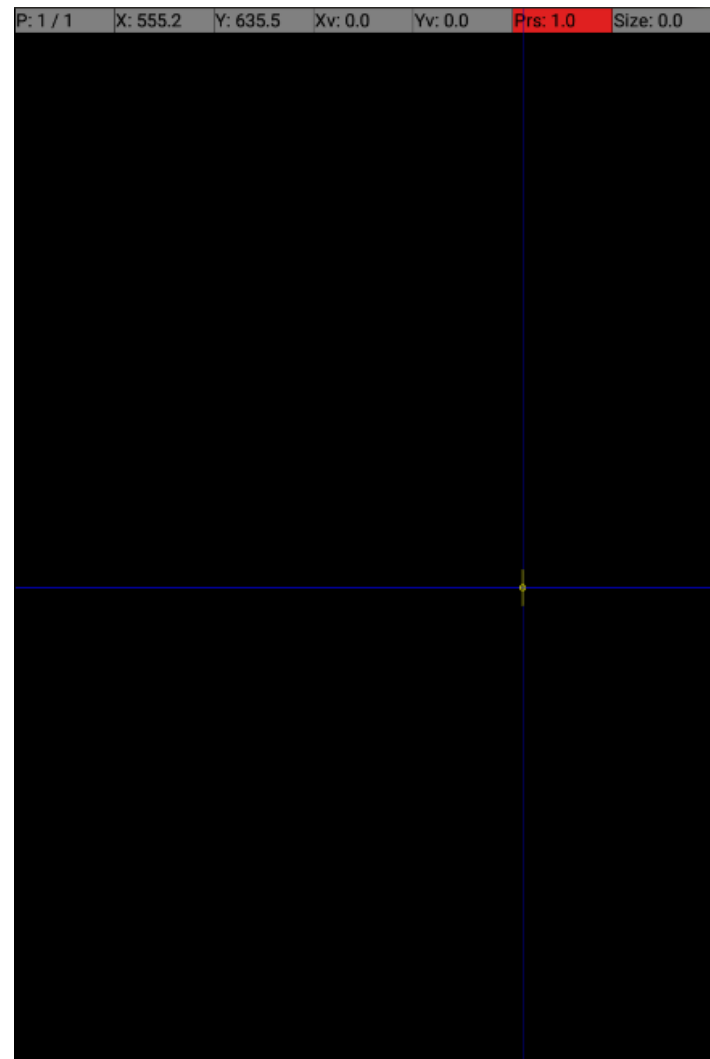


# 4.4 程序调试

- **4.4.2 DevTools**

- Pointer Location

- Pointer Location是屏幕点位置查看工具, 能够显示触摸点的X轴坐标和Y轴坐标
    - Pointer Location的使用画面



## 4.4 程序调试

- **4.4.2 DevTools**

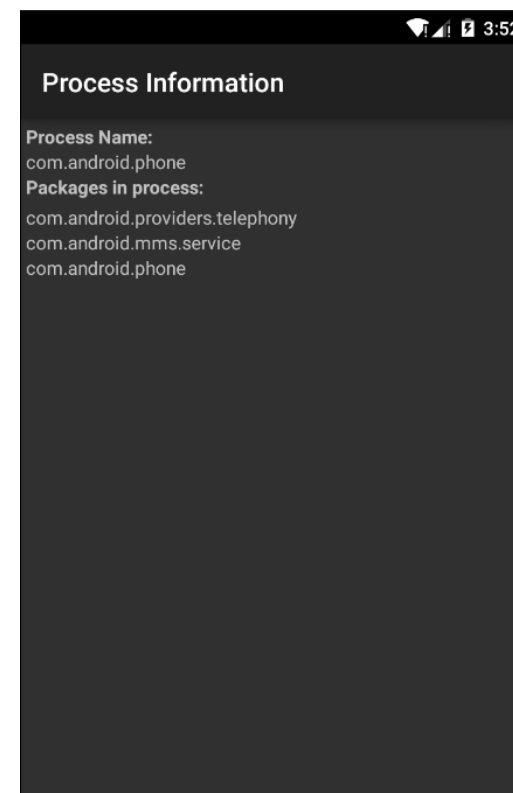
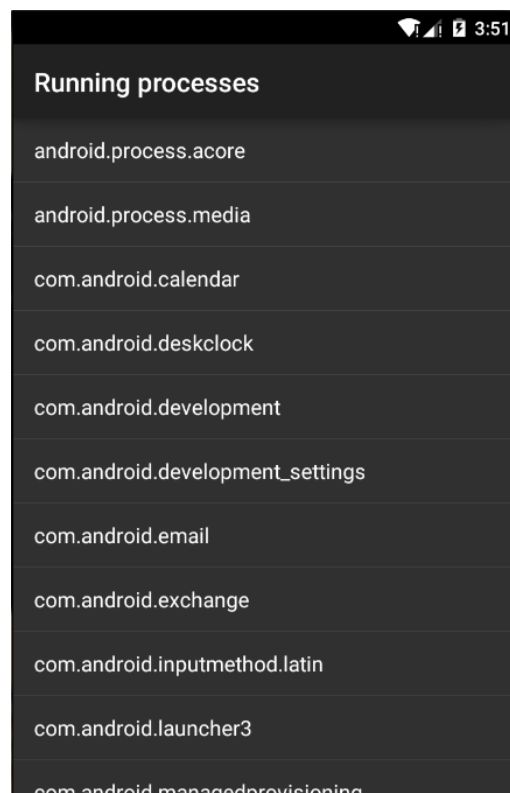
- Running processes

- Running processes能够查看在Android系统中正在运行的进程，并能查看进程的详细信息，包括进程名称和进程所调用的程序包
    - Andoird模拟器缺省情况下运行的进程和com.android.phone进程的详细信息

# 4.4 程序调试

- **4.4.2 DevTools**

- Running processes
  - Android模拟器所运行进程的列表和com.android.phone进程的详细信息

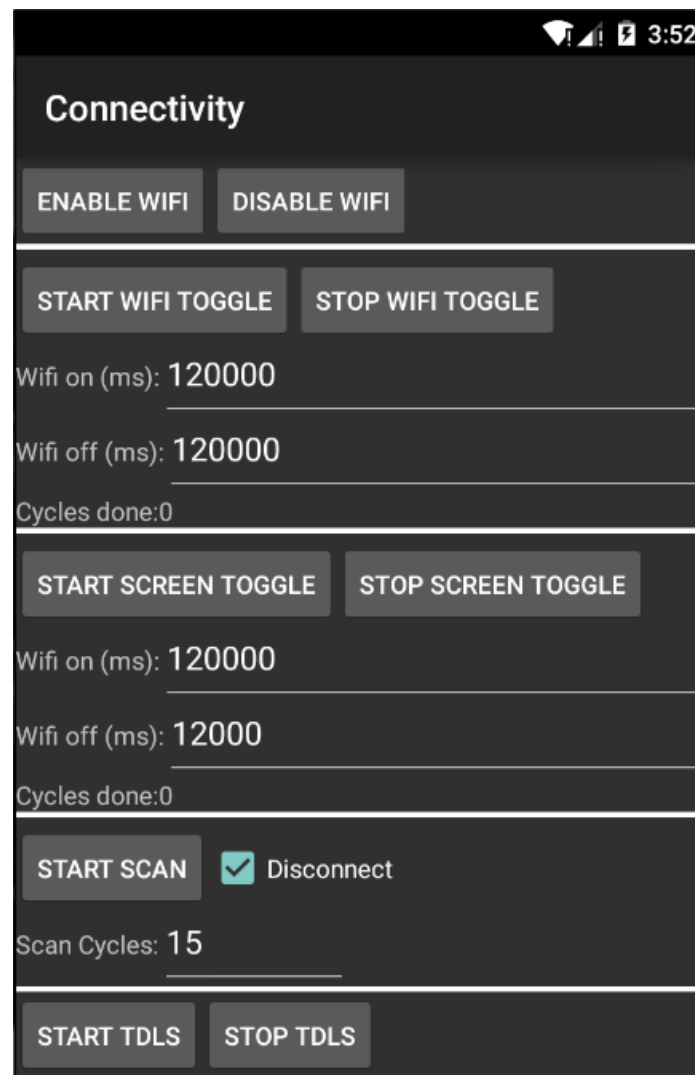


# 4.4 程序调试

- **4.4.2 DevTools**

- Connectivity

- Connectivity允许用户控制Wifi、屏幕锁定界面、MMS和导航开启与关闭，并可以设置Wifi和屏幕锁定界面的开启与关闭的周期

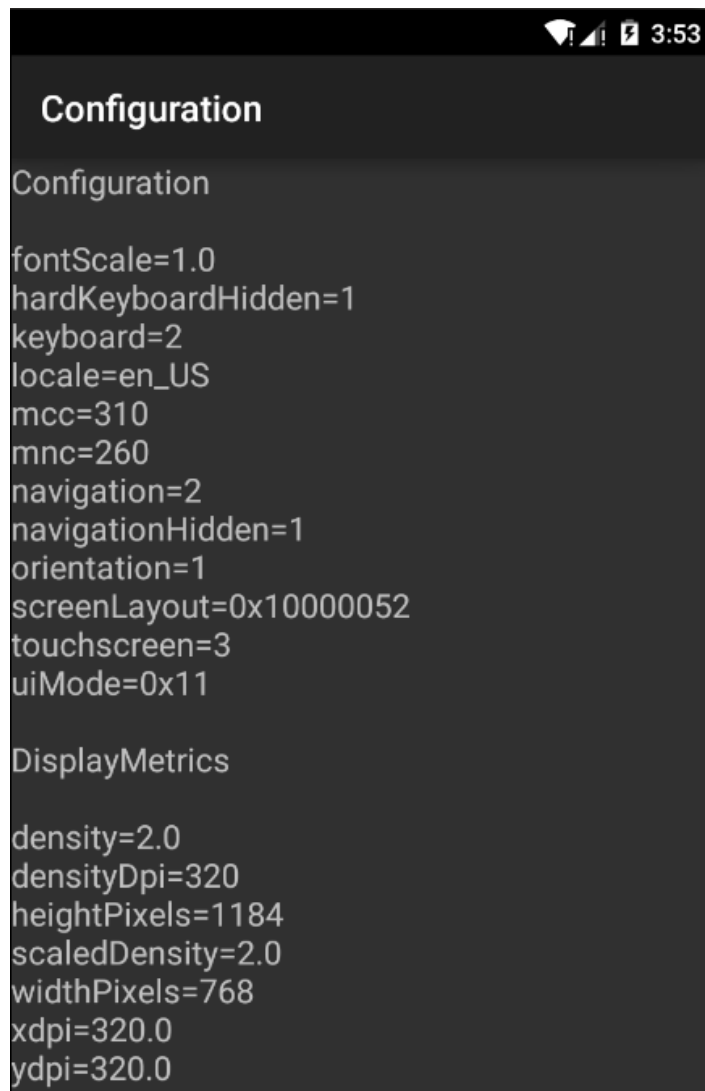


# 4.4 程序调试

- **4.4.2 DevTools**

- Configuration

- Configuration中详细列出了Android系统的配置信息，包括屏幕分辨率、字体缩放比例、屏幕初始方向、触屏类型、导航、本地语言和键盘等信息



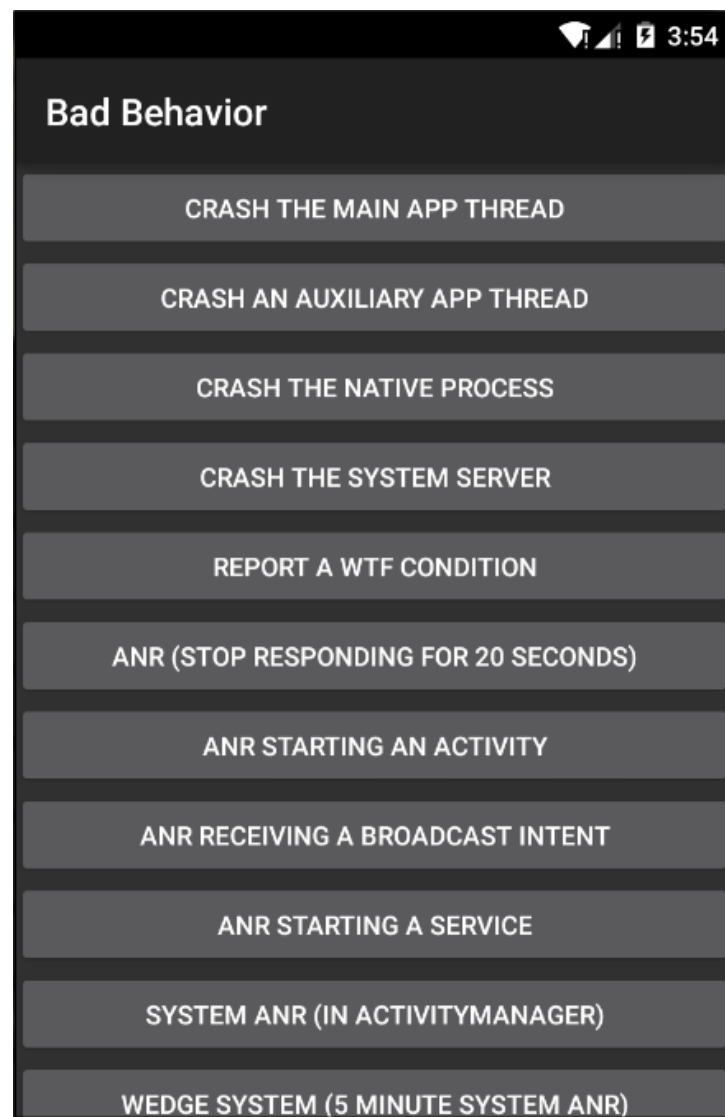


# 4.4 程序调试

- **4.4.2 DevTools**

- Bad Behavior

- Bad Behavior中可以模拟各种程序崩溃和失去响应情况，如主程序崩溃、系统服务崩溃、启动Service时失去响应和启动Activity时失去响应等



# 4.4 程序调试

- 4.4.2 DevTools

- Bad Behavior

- Bad Behavior选项

选项	说明
CRASH THE MAIN APP THREAD	应用程序主线程崩溃
CRASH AN AUXILIARY APP THREAD	应用程序工作线程崩溃
CRASH THE NATIVE PROCESS	本地进程崩溃
CRASH THE SYSTEM SERVER	系统服务器崩溃
REPORT A WTF CONDITION	报告WTF
ANR(STOP RESPONDING FOR 20 SECONDS)	应用程序无响应（Application Not Responding, ANR）20秒
ANR STARTING AN ACTIVITY	启动Activity时应用程序无响应
ANR STARTING A BROADCAST INTENT	启动Intent时应用程序无响应
ANR STARTING A SERVICE	启动Service时应用程序无响应
SYSTEM ANR (IN ACTIVITY MANAGER)	Activity管理器级别ANR
WEDGE SYSTEM (5 MINITE SYSTEM ANR)	Wedge在5分钟内无响应



## 习题：

- 1.简述Android系统前台进程、可见进程、服务进程、后台进程和空进程的优先级排序原因。
- 2.简述Android系统的四种基本组件Activity、Service、BroadcastReceiver和ContentProvider的用途。
- 3.简述Activity生命周期的四种状态，以及状态之间的变换关系。
- 4.简述Activity事件回调函数的作用和调用顺序。



THANKS

谢 谢 观 看