# 一、Settings 源码分析

一、Settings 5.1 源码分析

1.1、

# 二、Service

1. START\_STICKY\_COMPATIBILITY 与START\_STICKY

1):START\_STICKY：如果service进程被kill掉，保留service的状态为开始状态，但不保留递送的intent对象。随后系统会尝试重新创建service，由于服务状态为开始状态，所以创建服务后一定会调用onStartCommand(Intent,int,int)方法。如果在此期间没有任何启动命令被传递到service，那么参数Intent将为null。

2):START\_NOT\_STICKY：“非粘性的”。使用这个返回值时，如果在执行完onStartCommand后，服务被异常kill掉，系统不会自动重启该服务

3):START\_REDELIVER\_INTENT：重传Intent。使用这个返回值时，如果在执行完onStartCommand后，服务被异常kill掉，系统会自动重启该服务，并将Intent的值传入。如果此服务的进程在启动时被终止（从onstartcommand（intent，int，int）返回之后），那么它将被安排重新启动，最后一次传递的intent将通过onstartcommand（intent，int，int）重新传递给它。在服务调用stopself（int）并将start id提供给onstartcommand（intent，int，int）之前，此意图将保持预定的重新传递。服务将不会接收带有空意图的OnStartCommand（intent，int，int）调用，因为只有在未完成对发送给它的所有意图的处理时，它才会重新启动（并且任何此类挂起事件都将在重新启动时传递）

4):START\_STICKY\_COMPATIBILITY：START\_STICKY的兼容版本，但不保证服务被kill后一定能重启。

2. startService和bindService混合使用分析

3、onRebind何时调用？有何作用？

三、IntentService

3.1、不应该重新实现onStartCommand,而是去复写onHandleIntent

3.2、IntentService默认只支持两种返回值START\_REDELIVER\_INTENT或者START\_NOT\_STICKY，并且由setIntentRedelivery方法决定，默认是START\_NOT\_STICKY,不重新创建。

四、服务 Service 清单文件中可设置的属性

五、服务 Service 不被kill

5.1、如果走生命周期，onDestory中发送广播，重启服务。

5.2、对于不走生命走起，直接kill进程的，如何保护。如小米手-最近任务-kill应用。

# 三、RecyclView

## 1、RecyclView局部更新

1.1、RecyclView性能优化-妙用payload(有效载荷) 针对局部更新

# 四、Camera模块

1、

# 五、Activity模块

一、启动模式

# 六、跨进程通信

6.1、早期版本的aidl，Messenger

6.2、aidl

# 七、网络通信或框架

7.1、tcp/udp

7.2、OKHttp

7.3、Retrofit

# 八、Handler

8.1、handler原理

8.2、HandlerThread使用及原理，源码分析。

# 九、java&android集合

9.1、集合可以在遍历的时候删除其中的元素吗？

9.2、Vector ，是线程安全的，源码中，方法有synchronized

final Vector<ActionBean> twoVector = new Vector<ActionBean>();

vector.addAll(twoVector);追加集合不起作用。

vector中只有新添加的twoVector，而原有的数据没有了。

9.3、CopyOnWriteArrayList

【A thread-safe random-access list.

\*

\* <p>Read operations (including {@link #get}) do not block and may overlap with

\* update operations. Reads reflect the results of the most recently completed

\* operations. Aggregate operations like {@link #addAll} and {@link #clear} are

\* atomic; they never expose an intermediate state.】

【线程安全随机访问列表。

<p>读取操作（包括@link get）不阻塞，可能与

更新操作。读取反映最近完成的结果

操作。聚合操作，如@link addall和@link clear

\*原子的；它们从不暴露中间状态。】

9.3.2、CopyOnWriteArrayList源码分析。

9.4、java中集合如List中遍历同时删除元素可以吗？如何实现。

原来java中的集合边遍历边删除是需要使用迭代器中的方法的

Iterator<String> it = list.iterator();

String str = it.next();

it.remove();

# 十、EventBus

10.1、evnetbus。又不知道到底哪用了

AS安装EventBus插件--EventBus事件总线专用的事件导航.

http://www.6maa.com/android/demo/7459.html