一**，启动循环线程**

1，定义要循环执行的线程，如下，该线程继承与ScheduleUtil.SRunnable接口，

getName代表线程的唯一名字，用于识别线程，不能为空。



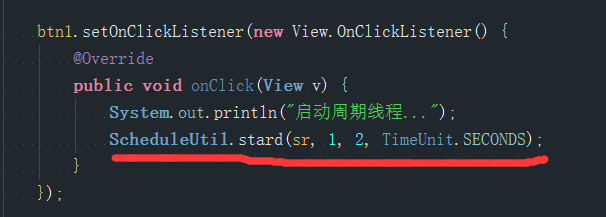
2，提交线程，即可执行，如下

参数一：代表执行的线程；

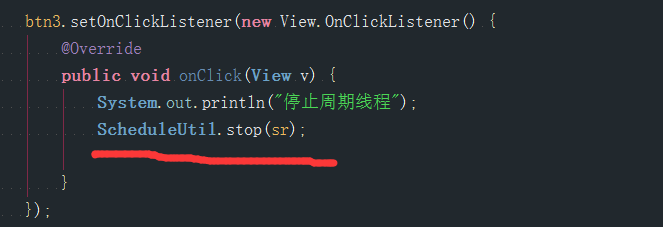
参数二：代表延迟执行的时间，单位/秒，代表提交线程后一秒后执行；

参数三：代表执行周期时间，单位/秒；代表每两秒执行一次；

参数四：代表时间单位；固定TimeUnit.SECONDS；

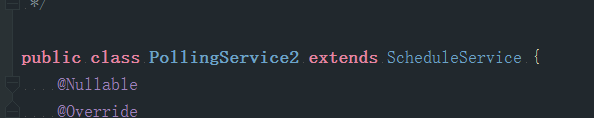


3，停止线程，如下：



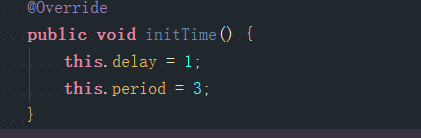
1. **启动后台服务循环线程**

1，创建服务类，继承 ScheduleService，如下



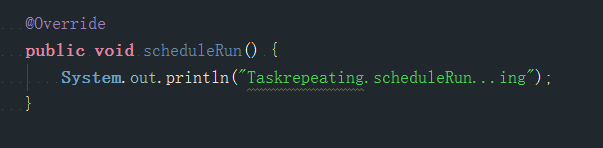
该类有5个实现方法；

1. public IBinder onBind(Intent intent) {};//继承自Service，用于服务绑定；
2. public void create() {};//对应onCreate()方法；
3. public void startCommand(Intent intent, int flags, int startId) {};//对应onStartCommand方法；
4. public void initTime() {}//用于初始化执行延迟时间、执行周期时间，如下



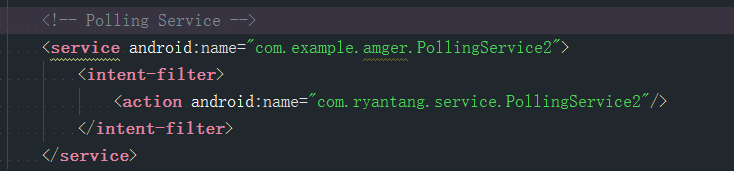
延迟一秒执行，每3秒执行；

E. public void scheduleRun() {};//需要执行的动作，该动作执行于子线程中无需另起线程，如下，每3秒中打印一次 Taskrepeating.scheduleRun...ing；



2，在manifests中注册该服务，如下：

其中的AntionName用于intent跳转;

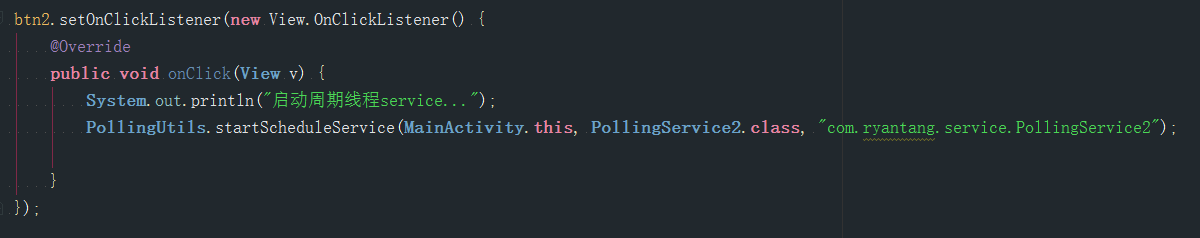


3,启动服务

参数一，context；

参数二，要启动的服务类；

参数三，在manifest中注册的ActionName；



4，停止服务

参数一，context；

参数二，要启动的服务类；

参数三，在manifest中注册的ActionName；

