转载请注明链接:https://blog.csdn.net/feather_wch/article/details/82557043

> 详细介绍IntentService，并解析源码。看完本文你可以掌握问题汇总中所有问题的答案。做到精通IntentService

# IntentService

版本：2018/9/9-1(10:10)



# 问题汇总

> 1. 总结IntentService相关所有可能的问题，用于自我检测和查漏补缺。
> 2. 【☆】标记的是补充问题，直接给答案。其余问题答案都在文中。

这些问题你会吗?

1. IntentService是什么?
2. IntentService的执行方式是串行还是并行?
3. IntentService可以执行大量的耗时操作?
4. IntentService和Service的区别
5. IntentService的优先级和Thread谁高?
6. IntentService的使用步骤?
7. IntentService的onCreate底层原理
8. IntentService的SerivceHandler的作用?
9. IntentService内部去停止Service为什么不直接采用stopSelf()?
10. IntentService是如何停止HandlerThread的Looper消息循环的?
11. IntentService多次startService会多次回调onHandleIntent()的内部流程?
12. 【☆】IntentService的消息处理流程
    > 通过handler投递到内部HandlerThread的消息队列中，取出并且执行ServiceHandler.handleMessage()，最后执行onHandleIntent()

# IntentService(5)

1、IntentService是什么?

> 1. 一个封装了 `HandlerThread` 和 `Handler` 的异步框架。
> 2. 是一种特殊 `Service` ， 继承自Service，是 抽象类 ,必须创建 子类 才可以使用。
> 3. 可用于执行后台耗时的任务，任务执行后会 自动停止
> 4. 具有 高优先级 (服务的原因),优先级比单纯的 线程 高很多，适合 高优先级 的后台任务，且不容易被系统杀死。
> 5. 启动方式和Service一样。
> 6. 可以多次启动，每个耗时操作都会以工作队列的方式在IntentService的onHandleIntent回调方法中执行。
> 7. 串行执行。

2、IntentService的执行方式是串行还是并行?

> 串行

3、IntentService可以执行大量的耗时操作?

> 1. 如果只有一个任务，是可以进行耗时操作的。
> 2. 如果有很多任务，由于内部的HandlerThread是串行执行任务，会导致耗时操作阻塞了后续任务的执行。

4、IntentService和Service的区别

> 1. 继承自Service
> 2. IntentService任务执行完后会自动停止
> 3. IntentService和Service优先级一致，比Thread高。
> 4. Service处于主线程不能直接进行耗时操作; IntentService内部有HandlerThread，可以进行耗时操作。

# 基本使用

5、IntentService的使用:

> 1-自定义LocalIntentService继承自IntentService
> 2-实现onHandleIntent(), 用于实现任务逻辑。

```
public class LocalIntentService extends IntentService{
    public LocalIntentService(String name) {
        super(name);
    }
    @Override
    protected void onHandleIntent(@Nullable Intent intent) {
        String action = intent.getStringExtra("task_action");
        Log.d("IntentService", "receive task :" + action);
        SystemClock.sleep(3000); //即使第一个任务休眠，后续的任务也会等待其执行完毕
        if("com.example.action.TASK1".equals(action)){
            Log.d("IntentService", "handle task :" + action);
        }
    }
}
```

> 3-开启IntentService

```
Intent service = new Intent(this, LocalIntentService.class);
service.putExtra("tast_action", "com.example.action.TASK1");
startService(service);

service.putExtra("tast_action", "com.example.action.TASK2");
startService(service);

service.putExtra("tast_action", "com.example.action.TASK3");
startService(service);
```

# 源码和原理机制(6)

1、IntentService的onCreate底层原理

> 1. 构造了HandlerThread
> 2. 并在每部保存了HandlerThread的Looper
> 3. 并且使用该Looper创建了ServiceHandler

```
    //IntentService第一次启动调用
    public void onCreate() {
        super.onCreate();
        //1. 创建一个HanlderThread
        HandlerThread thread = new HandlerThread("IntentService[" + mName + "]");
        thread.start();
        //2. 通过HanlderThread的Looper来构建Handler对象mServiceHandler
        mServiceLooper = thread.getLooper();
        mServiceHandler = new ServiceHandler(mServiceLooper);
    }
```

2、IntentService的SerivceHandler

1. HandlerThread会串行的取出任务并且执行，会调用ServiceHandler的handleMessage去处理任务。
2. handlerMessage会去调用我们自定义的 `onHandleIntent`
3. 任务执行完毕后通过 `stopSelf(startId)` 停止Service。
4. 任务结束后，在onDestory()中会退出HandlerThread中Looper的循环。

```
//ServiceHandler接收并处理onStart()方法中发送的Msg
private final class ServiceHandler extends Handler {
    public ServiceHandler(Looper looper) {
        super(looper);
    }
    @Override
    public void handleMessage(Message msg) {
        //1. 直接在onHandleIntent中处理(由子类实现)
        onHandleIntent((Intent)msg.obj);
        /**================================================
         * 2. 尝试停止服务(会等待所有消息都处理完毕后，才会停止)
         *    不能采用stopSelf()—会立即停止服务
         *================================================*/
        stopSelf(msg.arg1); //会判断启动服务次数是否与startId相等
    }
}

public void onDestroy() {
    mServiceLooper.quit();
}//销毁时会停止looper
```

3、IntentService内部去停止Service为什么不直接采用stopSelf()?

1. 采用stopSelf()——会立即停止服务
2. 采用stopSelf(startId)，会等所有消息全部处理完毕后，才会停止。 会判断启动服务次数是否与`startId`相等

4、IntentService是如何停止HandlerThread的Looper消息循环的?

1. 调用stopSelf(startId)后。
2. 任务全部执行完，会停止服务，并且回调onDestory()。调用Looper的quit()方法即可

5、IntentService多次startService会多次回调onHandleIntent()的内部流程?

1. startService()->onStartCommand()->onStart()
2. 通过HandlerThread的handler去发送消息。
3. HandlerThread在处理任务时，会去调用onHandleIntent方法。

```
public abstract class IntentService extends Service {
        private volatile Looper mServiceLooper;
        private volatile ServiceHandler mServiceHandler;
        ...省略...

        //IntentService每次启动都会调用
        public int onStartCommand(Intent intent, int flags, int startId) {
            //3. 直接调用onStart
            onStart(intent, startId);
            return mRedelivery ? START_REDELIVER_INTENT : START_NOT_STICKY;
        }
        public void onStart(Intent intent, int startId) {
            //4. 通过mServiceHandler发送一个消息(该消息会在HanlderThread中处理)
            Message msg = mServiceHandler.obtainMessage();
            msg.arg1 = startId;
            msg.obj = intent;
            mServiceHandler.sendMessage(msg);
        }
        //2. 该Intent与startService(intent)中的Intent完全一致
        protected abstract void onHandleIntent(Intent intent);

}
```

### 6、IntentService的源码要点总结

> 1. `IntentService` 通过发送消息的方式向 `HandlerThread` 请求执行任务
> 2. `HandlerThread` 中的 `looper` 是顺序处理消息，因此有多个后台任务时，都会按照外界发起的顺序 排队执行
> 3. 启动流程：onCreate()->onStartCommand()->onStart()
> 4. 消息处理流程：ServiceHandler.handleMessage()->onHandleIntent()

# 参考资料


1. HandlerThread详解