**Android HandlerThread详解**

[朋永](https://www.jianshu.com/u/a0a26f5dabd2)关注

0.292017.07.20 15:30:23字数 1,167阅读 7,610

转载请注明出处：[http://www.jianshu.com/p/9295d31ea08a](https://www.jianshu.com/p/9295d31ea08a)

对于Handler不太懂的可以参考我的这两篇文章：  
[Android Handler的基本使用]  
[深入理解Handler、Looper、Messagequeue]  
这篇主要说一下HandlerThread的使用方法，及分析下其源码。

**1 HandlerThread##**

HandlerThread类介绍

Handy class for starting a new thread that has a looper. The looper can then be used to create handler classes. Note that start() must still be called.

HandlerThread是Android API提供的一个方便、便捷的类，使用它我们可以快速的创建一个带有Looper的线程。Looper可以用来创建Handler实例。<font color=red>**注意：**</font>start()仍然必须被调用。

如下是HandlerThread使用的demo。

**package com.zpengyong.hand;**

**import android.app.Activity;**

**import android.os.Bundle;**

**import android.os.Handler;**

**import android.os.HandlerThread;**

**import android.os.Looper;**

**import android.os.Message;**

**import android.util.Log;**

**import android.view.View;**

**import android.view.View.OnClickListener;**

**import android.widget.Button;**

**import android.widget.TextView;**

**public class MainActivity extends Activity {**

**private final static String TAG = "MainActivity";**

**private Button mGet;**

**private TextView mResult;**

**protected final int MSG\_GET = 1;**

**protected final int MSG\_RESULT = 2;**

**private HandlerThread mHandlerThread;**

**//子线程中的Handler实例。**

**private Handler mSubThreadHandler;**

**//与Ui线程绑定的Handler实例。**

**private Handler mUiHandler = new Handler(){**

**public void handleMessage(Message msg) {**

**Log.i(TAG, "mUiHandler handleMessage thread:"+Thread.currentThread());**

**switch (msg.what) {**

**case MSG\_RESULT:**

**mResult.setText((String)msg.obj);**

**break;**

**default:**

**break;**

**}**

**};**

**};**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**setContentView(R.layout.activity\_main);**

**Log.i(TAG, "onCreate thread:"+Thread.currentThread());**

**mGet = (Button) findViewById(R.id.get);**

**mGet.setOnClickListener(new OnClickListener() {**

**@Override**

**public void onClick(View v) {**

**mSubThreadHandler.sendEmptyMessage(MSG\_GET);**

**}**

**});**

**mResult = (TextView) findViewById(R.id.result);**

**initHandlerThraed();**

**}**

**private void initHandlerThraed() {**

**//创建HandlerThread实例**

**mHandlerThread = new HandlerThread("handler\_thread");**

**//开始运行线程**

**mHandlerThread.start();**

**//获取HandlerThread线程中的Looper实例**

**Looper loop = mHandlerThread.getLooper();**

**//创建Handler与该线程绑定。**

**mSubThreadHandler = new Handler(loop){**

**public void handleMessage(Message msg) {**

**Log.i(TAG, "mSubThreadHandler handleMessage thread:"+Thread.currentThread());**

**switch(msg.what){**

**case MSG\_GET:**

**try { //模拟延时处理**

**Thread.sleep(1000);**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**double number = Math.random();**

**String result = "number："+number;**

**//向ui线程发送消息，更新ui。**

**Message message = new Message();**

**message.what = MSG\_RESULT;**

**message.obj = result;**

**mUiHandler.sendMessage(message);**

**break;**

**default:**

**break;**

**}**

**};**

**};**

**}**

**@Override**

**protected void onDestroy() {**

**super.onDestroy();**

**Log.i(TAG, "onDestroy");**

**//退出HandlerThread的Looper循环。**

**mHandlerThread.quit();**

**}**

**}**

上述代码比较简单，功能也比较简单，可以在此基础上进行扩展。  
在Actvitiy创建的时候调用initHandlerThraed()函数：

1. 创建HandlerThread线程
2. 运行线程
3. 获取HandlerThread线程中的Looper实例
4. 通过Looper实例创建Handler实例，从而使mSubThreadHandler与该线程连接到一起。

多次点击按钮，打印信息如下所示：

07-13 05:15:07.662: I/MainActivity(1472): onCreate thread:Thread[main,5,main]

07-13 05:15:45.382: I/MainActivity(1472): mSubThreadHandler handleMessage thread:Thread[handler\_thread,5,main]

07-13 05:15:46.402: I/MainActivity(1472): mUiHandler handleMessage thread:Thread[main,5,main]

07-13 05:15:46.412: I/MainActivity(1472): mSubThreadHandler handleMessage thread:Thread[handler\_thread,5,main]

07-13 05:15:47.412: I/MainActivity(1472): mUiHandler handleMessage thread:Thread[main,5,main]

.....

点击按钮，向mSubThreadHandler发送消息，mSubThreadHandler中接收到消息进行处理，由打印可知mSubThreadHandler的handleMessage方法运行在子线程中。  
模拟耗时操作，生成随机数，然后向主线程中（mUiHandler）发送消息（Message）。

mUiHandler的handleMessage方法运行在主线程，可以用来更新Ui界面。

Activity销毁的时候，调用mHandlerThread.quit()，退出HandlerThread的Looper循环。

效果图如下：

效果

**2 HandlerThread源码分析**

源码路径路径：frameworks/base/core/java/android/os/HandlerThread.java

先看下HandlerThread的构造方法。

**public class HandlerThread extends Thread {**

**int mPriority;**

**int mTid = -1;**

**Looper mLooper;**

**//@param name 线程名**

**public HandlerThread(String name) {**

**super(name);**

**mPriority = Process.THREAD\_PRIORITY\_DEFAULT;**

**}**

**/\*\***

**\* Constructs a HandlerThread.**

**\* @param name**

**\* @param priority The priority to run the thread at. The value supplied must be from**

**\* {@link android.os.Process} and not from java.lang.Thread.**

**\*/**

**public HandlerThread(String name, int priority) {**

**super(name);**

**mPriority = priority;**

**}**

**。。。。**

HandlerThread是Thread（线程）的子类。创建一个HandlerThread实例，也就是创建了一个特殊的线程实例。  
HandlerThread提供了两个构造方法：

* HandlerThread(String name) 参数为线程名称，线程优先级为Process.THREAD\_PRIORITY\_DEFAULT。
* HandlerThread(String name, int priority)，name为线程名称，priority为设置的线程优先级。

我们知道线程需要通过start()方法来运行线程，HandlerThread也是这样的。接着看下线程运行的run()方法。

**/\*\***

**\* Call back method that can be explicitly overridden if needed to execute some**

**\* setup before Looper loops.**

**\*/**

**protected void onLooperPrepared() {**

**}**

**@Override**

**public void run() {**

**//获取进程id**

**mTid = Process.myTid();**

**//创建Looper实例**

**Looper.prepare();**

**synchronized (this) {**

**//获取当前线程的Looper实例**

**mLooper = Looper.myLooper();**

**notifyAll();**

**}**

**//设置线程优先级**

**Process.setThreadPriority(mPriority);**

**onLooperPrepared();**

**//开始循环**

**Looper.loop();**

**mTid = -1;**

**}**

由run方法可知HandlerThrea线程运行创建了Looper实例，并开启了Looper循环，循环从消息队列中获取消息并给Handler进行处理。对于Looper不太明白的可以参考这篇[深入理解Handler、Looper、Messagequeue](https://links.jianshu.com/go?to=http%3A%2F%2Fblog.csdn.net%2Fvnanyesheshou%2Farticle%2Fdetails%2F73484527)

onLooperPrepared()在Looper循环之前调用，如果需要在Looper循环之前执行一些设置，可以显式覆盖此方法。

接着看获取Looper实例

**//获取HandlerThread线程中的Looper实例**

**Looper loop = mHandlerThread.getLooper();**

对应源码：

**//此方法返回与此线程关联的Looper。 如果此线程未启动或由于任何原因isAlive（）返回false，此方法将返回null。**

**public Looper getLooper() {**

**if (!isAlive()) {**

**return null;**

**}**

**// 如果这个线程已经启动，将会被阻塞，直到mLooper被初始化为止。**

**synchronized (this) {**

**while (isAlive() && mLooper == null) {**

**try {**

**wait();**

**} catch (InterruptedException e) {**

**}**

**}**

**}**

**return mLooper;**

**}**

mHandlerThread.getLooper()获取与该线程绑定的Looper实例。mLooper是在HandlerThread的run()方法中赋值的（也就是在子线程中），getLooper是我们在主线程中调用，该方法会阻塞直到mLooper赋值。

然后demo中通过该looper实例创建Handler

**//创建Handler与该线程绑定。**

**mSubThreadHandler = new Handler(loop)**

你可能会好奇为什么要这样长久Handler而不是“new Handler()“这样呢？因为我们要创建的Handler要与子线程绑定到一起，要处理子线程中的消息，所以要通过子线程中的looper(有线程对应的消息队列)实例创建Handler。这样通过mSubThreadHandler发送的消息会添加到子线程中的消息队列中，然后Looper实例消息进行分发，交给mSubThreadHandler进行处理。

HandlerThread提供的线程退出方法：

**public boolean quit() {**

**Looper looper = getLooper();**

**if (looper != null) {**

**looper.quit();**

**return true;**

**}**

**return false;**

**}**

**public boolean quitSafely() {**

**Looper looper = getLooper();**

**if (looper != null) {**

**looper.quitSafely();**

**return true;**

**}**

**return false;**

**}**

quit和quitSafely都是退出HandlerThread的消息循环。其分别调用Looper的quit和quitSafely方法。  
quit方法会将消息队列中的所有消息移除（延迟消息和非延迟消息）。  
quitSafely会将消息队列所有的延迟消息移除，非延迟消息派发出去让Handler去处理。quitSafely相比于quit方法安全之处在于清空消息之前会派发所有的非延迟消息

HandlerThread适合处理本地IO读写操作（数据库，文件），因为本地IO操作大多数的耗时属于毫秒级别，对于单线程 + 异步队列的形式 不会产生较大的阻塞。而网络操作相对比较耗时，容易阻塞后面的请求，因此在这个HandlerThread中不适合加入网络操作。

至此HandlerThread就说完了。有什么问题欢迎大家指正、交流。