[](https://www.csdn.net/)

* [博客](https://blog.csdn.net/)
* [课程](https://edu.csdn.net/)
* [文库·商城](https://mall.csdn.net/)
* [问答](https://ask.csdn.net/)
* [社区](https://bbs.csdn.net/)
* [插件](https://plugin.csdn.net/?utm_source=toolbar)
* [认证](https://ac.csdn.net/landing1/?utm_source=crz_pc_toolbar1)
* [开源](https://git.csdn.net/)

 搜索

[登录/注册](https://passport.csdn.net/account/login)

[会员中心 [https://img-home.csdnimg.cn/images/20220208061453.gif](https://mall.csdn.net/vip)](https://mall.csdn.net/vip)

[收藏](https://i.csdn.net/#/user-center/collection-list?type=1)

[动态](https://blink.csdn.net/)

[创作](https://mp.csdn.net/)

**Android使用Handler实现子线程与子线程、主线程之间通信**

今天这篇文章只讲一下怎么使用[Handler](https://so.csdn.net/so/search?q=Handler&spm=1001.2101.3001.7020)实现子线程与子线程之间、子线程与主线程之间如何进行通信，关于具体的内部实现因为我也没研究过，所以这篇文章讲不了。

**一、子线程向主线程传值：**

这个实现比较简单，因为主线程自带Looper机制，所有我们不用创建Looper了，看一下代码吧：

首选在主线程里创建一个Handler

Handler mHandler = new Handler(){

@Override

public void handleMessage(Message msg) {

super.handleMessage(msg);

switch (msg.what) {

case 0:

//do something,refresh UI;

break;

default:

break;

}

}

};

然后开启一个子线程，在子线程里直接使用Handler发送消息即可

new Thread() {

public void run() {

Message message = new Message();

message.obj = "子线程发送的消息Hi~Hi";

mHandler .sendMessage(message);

};

}.start();

**二、主线程向子线程里发送消息：**

主线程向子线程发送消息的话，我们需要在子线程里初始化Looper，并在主线程里创建的Handler引用子线程的Looper（Handler中引用的是哪个线程的Looper，就在哪个线程里处理消息），下面看代码：

public class ThreadHandlerActivity extends Activity{

*//创建子线程*

class MyThread extends Thread{

private Looper looper;*//取出该子线程的Looper*

public void run() {

Looper.prepare();*//创建该子线程的Looper*

looper = Looper.myLooper();*//取出该子线程的Looper*

Looper.loop();*//只要调用了该方法才能不断循环取出消息*

}

}

private Handler mHandler;*//将mHandler指定轮询的Looper*

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

thread = new MyThread();

thread.start();*//千万别忘记开启这个线程*

*//下面是主线程发送消息*

mHandler = new Handler(thread.looper){

public void handleMessage(android.os.Message msg) {

Log.d("当前子线程是----->",Thread.currentThread()+"");

};

};

mHandler.sendEmptyMessage(1);

}

}

其实这样就可以达到主线程向子线程发送消息了，然而当我们运行后发现程序会Crash掉，报了一个控制针，这是因为在Handler初始化的时候，thread.looper还没有初始化，所以会报控制针，这时我们可以让主线程等待一下子线程，也可以来一个while循环来判断thread.looper是否初始化完成。不过Android本身还提供了一个方法，那就是**HandlerThread**：

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

tv = new TextView(this);

tv.setText("Handler实验");

setContentView(tv);

*//实例化一个特殊的线程HandlerThread，必须给其指定一个名字*

HandlerThread thread = new HandlerThread("handler thread");

thread.start();*//千万不要忘记开启这个线程*

*//将mHandler与thread相关联*

mHandler = new Handler(thread.getLooper()){

public void handleMessage(android.os.Message msg) {

Log.d("当前子线程是----->", Thread.currentThread()+"");

};

};

mHandler.sendEmptyMessage(1);*//发送消息*

}

这时HandlerMessage所在的线程就是HandlerThread 的子线程。  
然而HandlerThread 所创建处理的子线程里是不能重写Run（）方法的，你写了以后，会发现，HandlerMessage不执行了，这时因为HandlerMessage本身实现了Run（）方法，我们看一下内部实现：

@Override

public void run() {

mTid = Process.myTid();

Looper.prepare();

synchronized (this) {

mLooper = Looper.myLooper();

notifyAll();

}

Process.setThreadPriority(mPriority);

onLooperPrepared();

Looper.loop();

mTid = -1;

}

在源代码的第4行，进行了实例化自己的Looper，如果继续追踪源代码翻看其getLooper方法你会发现，如果一个Handler在与HandlerThread进行绑定时，发现Looper为空，Handler则会一直等待直到Looper被创建出来为止，然后才继续执行后续的代码。所以我们重写了HandlerThread的run方法，肯定就不会去创建Looper对象，那么绑定的Handler就会永远处于等待状态，自然而然就不会执行到HandlerMessage信息了。这也是为什么我们要使用HandlerThread这个特殊的线程，因为使用这个，我们不必关心多线程会混乱，Looper会为空等一系列问题，只要去关心我们要实现的逻辑就行了。

**三、子线程和子线程之间通信：**

其实子线程向子线程之间通信，其实就是在一个子线程中创建一个Handler，它的回调自然就在此子线程中，然后在另一个子线程中调用此handler来发送消息就可以了，不过记得写上Looper哦，下面看代码：

new Thread(new Runnable() {

@Override

public void run() {

String msg;

Looper.prepare();

childHandler = new Handler() {

@Override

public void handleMessage(Message msg) {

super.handleMessage(msg);

System.out.println("这个消息是从-->>" + msg.obj+ "过来的，在" + "btn的子线程当中" + "中执行的");

}

};

Looper.loop();*//开始轮循*

}

}).start();

其中 Looper.prepare()和Looper.loop()一定不要忘了写。

然后我们创建第二个子线程

new Thread(new Runnable() {

@Override

public void run() {

Looper loop = Looper.myLooper();

Message msg = childHandler.obtainMessage();

msg.obj = "btn2当中子线程";

childHandler.sendMessage(msg);

}

}).start();