[Android单线程模型中Message、Handler、Message Queue、Looper之间的关系](http://blog.csdn.net/andie_guo/article/details/9281569)

#### 四者之间的关系：

Handler获取当前线程中的looper对象，looper用来从存放Message的MessageQueue中取出Message，再由Handler进行Message的分发和处理.

#### 简单定义：

1、Message Queue(消息队列)：

用来存放通过Handler发布的消息，通常附属于某一个创建它的线程，可以通过Looper.myQueue()得到当前线程的消息队列.   
2、Handler：

可以发布或者处理一个消息或者操作一个Runnable，通过Handler发布消息，消息将只会发送到与它关联的消息队列，然也只能处理该消息队列中的消息.

3、Looper：

是Handler和消息队列之间通讯桥梁，程序组件首先通过Handler把消息传递给Looper，Looper把消息放入队列。Looper也把消息队列里的消息~~广播给所有的Handler~~：Handler接受到消息后调用handleMessage进行处理.  
4、Message：

消息的类型，在Handler类中的handleMessage方法中得到单个的消息进行处理,在单线程模型下，为了线程通信问题，Android设计了一个Message Queue(消息队列)， 线程间可以通过该Message Queue并结合Handler和Looper组件进行信息交换。

#### 详细介绍：

1. Message

Message消息，理解为线程间交流的信息，处理数据后台线程需要更新UI，则发送Message内含一些数据给UI线程。

2. Handler

Handler处理者，是Message的主要处理者，负责Message的发送，Message内容的执行处理。**后台线程就是通过传进来的Handler对象引用来sendMessage(Message)。**而使用Handler，需要implement 该类的 handleMessage(Message)方法，它是处理这些Message的操作内容，例如Update UI。通常需要子类化Handler来实现handleMessage方法。

3. Message Queue

Message Queue消息队列，用来存放通过Handler发布的消息，按照先进先出执行。

每个message queue都会有一个对应的Handler。Handler会向message queue通过两种方法发送消息：sendMessage或post。这两种消息都会插在message queue队尾并按先进先出执行。但通过这两种方法发送的消息执行的方式略有不同：通过sendMessage发送的是一个message对象,会被 Handler的handleMessage()函数处理；而通过post方法发送的是一个runnable对象，则会自己执行。

4. Looper

Looper是每条线程里的Message Queue的管家。Android没有Global的Message Queue，而Android会自动替主线程(UI线程)建立Message Queue，但在子线程里并没有建立Message Queue。所以调用Looper.getMainLooper()得到的主线程的Looper不为NULL，但调用Looper.myLooper() 得到当前线程的Looper就有可能为NULL。

判断Handler对象里面的Looper对象是属于哪条线程的，则由该线程来执行！

**1. 当Handler对象的构造函数的参数为空，则为当前所在线程的Looper；**

**2. Looper.getMainLooper()得到的是主线程的Looper对象，Looper.myLooper()得到的是当前线程的Looper对象。**

#### 源码：

#### **public** **class** MainActivity **extends** Activity **implements** OnClickListener {

#### **private** Button startbutton = **null**;

#### **private** TextView text = **null**;

#### **private** MyHandler mHandler = **null**;

#### **private** Thread thread;

#### @Override

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

#### **super**.onCreate(savedInstanceState);

#### setContentView(R.layout.main);

#### startbutton = (Button) findViewById(R.id.StartButton);

#### startbutton.setOnClickListener(**this**);

#### text = (TextView) findViewById(R.id.content);

#### }

#### @Override

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

#### **switch** (v.getId()) {

#### **case** R.id.StartButton:

#### // 每次单击都会建立一个新的线程，这肯定是不合适的

#### thread = **new** MyThread();

#### thread.start();// 开启一个子线程

#### **break**;

#### }

#### }

#### **private** **class** MyHandler **extends** Handler {

#### **public** MyHandler(Looper looper) {

#### **super**(looper);

#### }

#### @Override

#### **public** **void** handleMessage(Message msg) { // 处理消息

#### System.out.println("handle--id-->" + Thread.currentThread().getId());// 1

#### System.out.println("handle--name-->" + Thread.currentThread().getName());// main

#### text.setText(msg.obj.toString());// 此时handle在主线程当中，可以处理界面更新

#### }

#### }

#### /\*\*

#### \* Android 会自动替主线程建立Message Queue ,在这个子线程里并没有建立Message Queue 。 所以，myLooper

#### \* 值为null ，而mainLooper 则指向主线程里的Looper 。于是，执行到： mHandler = new MyHandler

#### \* (mainLooper); 此mHandler属于主线程。 mHandler.sendMessage(m);

#### \*/

#### **private** **class** MyThread **extends** Thread {

#### /\*\*

#### \* Looper.myLooper();获得当前的Looper

#### \* Looper.getMainLooper ();获得UI线程的Lopper

#### \*/

#### @Override

#### **public** **void** run() {

#### System.out.println("MyThread------id------>" + Thread.currentThread().getId());

#### System.out.println("MyThread------name------>" + Thread.currentThread().getName());

#### Looper curLooper = Looper.myLooper();// MyThread线程

#### Looper mainLooper = Looper.getMainLooper();

#### String msg;

#### **if** (curLooper == **null**) {

#### // 把当前handler绑定在mainLooper线程上；此时mainLooper线程当中

#### mHandler = **new** MyHandler(mainLooper);

#### msg = "curLooper is null";

#### } **else** {

#### mHandler = **new** MyHandler(curLooper);// 将mHandler与curLooper绑定

#### msg = "This is curLooper";

#### }

#### mHandler.removeMessages(0);

#### Message m = mHandler.obtainMessage(1, 1, 1, msg);

#### // mHandler对象而将消息m传给curLooper，然后放入MessageQueue里。

#### // Looper对象看到MessageQueue里有消息m，就将它广播出去，mHandler对象接到此讯息时，会呼叫其handleMessage()函数来处理

#### mHandler.sendMessage(m);// 将消息添加到了curLooper相关联的消息队列中；

#### }

#### }

#### }

#### 结果验证：

