消息队列

Handler的消息传递机制

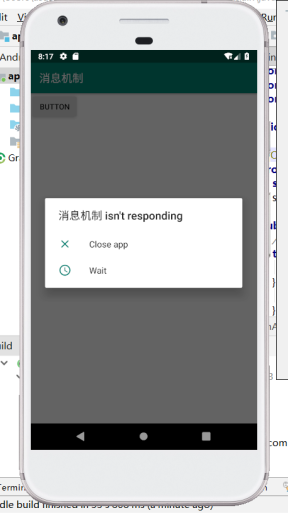
Handler的编程接口

为什么要使用线程？

当主线程执行的操作时间过长，就无法执行其他操作。消息队列中的消息无法处理，UI界面就不会更新，用户事件也不会处理。

如果主线程执行一个操作超过5s，Android可能会弹出一个应用程序没有响应的对话框，你的Activity可能会被杀掉。

所以当我们需要进行一个耗时的操作时，我们不能在主线程上来定义这个操作，而应该再开一个子线程来完成这个工作。



UI线程中的操作要尽量短小，费时的操作(如网络连接，数据库访问)等要在单独的线程中完成。

如果UI线程超过5s没有响应用户请求，会弹出ANR的对话框

常见的耗时的操作如下：

网络连接和网络数据的获取

对存储器的读写

大量数据计算

…

所以这些工作必须在子线程（工作线程）中来完成

那么Android中如何保证子线程和主线程之间能够进行一个相应的通讯呢？

Ans：Android的Activity中本身能够产生一个消息队列（Message Queue）保证线程间通信。

消息队列用来存放Handler发布的消息。Android在第一次启动程序时会默认为UI线程创建一个关联的消息队列，用来管理程序的组件。如Activity，Service，Broatcast Receiver等。这些都可以认为是UI线程

可以在工作线程中创建Handler与UI线程通信

消息队列是由Looper的类对象来产生的，即由消息循环来产生。然后我们再用到Handler，和它相对应的Looper所产生的消息队列去进行通信。

涉及到以下三个类：

Message类（消息类）

Looper类（消息循环类）

Handler类 当UI线程和子线程进行通信的情况下，一般来说都会通过Handler去完成



一个activity是我们的一个主线程，在它上面可以创建多个工作线程，我们将这些工作线程发送的消息都放到主线程中间，那么主线程就可以去做处理了。也就意味着我们可以在主线程中间来更新组件相应的属性。只需要把子线程发送的消息放进来就可以了。



（1）工作线程可以通过Handler对象和主线程通讯

（2）Handler对象的所有工作将在主线程中执行

（3）Handler类需要实现handleMessage()方法，来处理消息队列中取出的Message对象。

（4）handleMessage()方法由主线程调用，可以在需要的时候更新UI界面。但是必须确保此方法快速完成，因为其他UI操作会等待它完成才能执行。

（5）可以在Message中附加不同的参数

Handler的编程接口

void handleMessage(Message message)

接收消息（消息队列中的消息）

boolean sendEmptyMessage(int what)

发送只有一个what值的消息（工作线程/子线程）

boolean sendMessage(Message message)

发送消息（内含what以及参数arg1、arg2等）到Handler，在handleMessage中处理（工作线程/子线程）

boolean hasMessafe(int what)

判断是否有what值的消息

boolean post(Runnable r)

将一个Runnable添加到消息队列

实例代码：

布局文件：activity\_main.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**TextView  
 android:id="@+id/tvTest"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text=""** />  
  
  
 <**Button  
 android:id="@+id/button1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginTop="105dp"  
 android:onClick="test"  
 android:text="Button"** />  
  
 <**Button  
 android:id="@+id/btnAsync"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignBottom="@+id/button1"  
 android:layout\_alignParentStart="true"  
 android:layout\_marginStart="106dp"  
 android:layout\_marginBottom="-1dp"  
 android:onClick="test"  
 android:text="异步完成工作"** />  
</**RelativeLayout**>

后台事件处理程序：MainActivity.java

**package** com.amaker.hjs.thread;  
  
**import** android.os.Handler;  
**import** android.os.Message;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.TextView;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **private** TextView **tvTest**;  
 *//主线程创建Handler对象* **private** Handler **handler** = **new** Handler(){  
 @Override  
 **public void** handleMessage(Message msg) { *//参数msg是从主线程的消息队列中拿出的  
 //处理消息方法，该方法运行在主线程中，即哪一个Looper创建的就运行在哪里* **switch** (msg.**what**){  
 **case** 0x0001: *//what标识*  
 **int** index = msg.**arg1**;  
 **tvTest**.setText(index+**""**); *//这是允许的，因为这是运行在主线程中的* **break**;  
 }  
 };  
 };  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **this**.**tvTest**=(TextView)findViewById(R.id.***tvTest***);  
 }  
 **public void** test(View view){  
 *//按钮单击完成的工作  
/\* try {  
 Thread.sleep(20000);//睡眠20s，模拟一个非常耗时的工作  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }\*/* **switch** (view.getId()){  
 **case** R.id.***btnAsync*** :  
 *//使用子线程完成  
 //实现线程方式1：工作线程/子线程是一个Thread的匿名类，并重写run()方法* **new** Thread(){  
 @Override  
 **public void** run() {  
 **for** (**int** i =0 ; i<100 ; i++){  
 *//tvTest.setText(i+""); //这种做法不允许，因为子线程不可更新UI线程中控件的属性  
 //工作线程发送消息，是需要发送到主线程中间的，也就是将消息发送到主线程中间的Looper所产生的消息队列里，  
 // 因此我们的Handler对象就必须是主线程创建的* Message msg = **new** Message();*//子线程创建message* msg.**what**=0x0001 ; *//message必须要有标识what，由于子线程可以发送多个消息给主线程，因此what用于识别每一个子线程的不同消息* msg.**arg1**=i; *//还需要有参数arg。当传递整数的时候可以用arg1或arg2来传递，其他对象用obj进行替代* **handler**.sendMessage(msg);*//使用主线程的handler发送消息，才能送到对应于主线程Looper创建的消息队列中* **try** {  
 Thread.*sleep*(1000);*//由于for语句循环非常快，所以我们每一次循环休息1s降低速度* } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 }  
 }  
 }.start();  
 **break**;  
  
 }  
 }  
}