HandlerThread创建一个子线程，在UI线程中将一个handler对象绑定给HandlerThread创建的线程，覆写handler对象的handlemessage方法，该方法会在HandlerThread创建线程中执行。就可以实现在UI线程中控制子线程。

package com.example.ai.application.HandlerThreadAnnotation;  
  
import android.app.Activity;  
import android.os.Bundle;  
import android.os.Handler;  
import android.os.HandlerThread;  
import android.os.Message;  
import android.text.Html;  
import android.util.Log;  
import android.widget.TextView;  
  
import com.example.ai.application.R;  
  
*/\*\*  
 \* Cr  
 \*/*public class activity extends Activity {  
  
 private TextView mTvServiceInfo;  
  
 **private HandlerThread handlerThread;  
 private Handler handler;**  
 private static final int *MSG\_UPDATE\_INFO* = 0x110;  
  
 **//与UI线程管理的handler  
 private Handler UIHandler = new Handler(){  
 @Override  
 public void handleMessage(Message msg) {  
 //主线程接收到消息，并控制让与子线程绑定的handler给子线程自己发送了一条消息  
 Log.i("main", "UI-handleMessage: 主线程接收到消息，并控制让与子线程绑定的handler给子线程自己发送了一条消息");  
 handler.sendEmptyMessage(MSG\_UPDATE\_INFO);  
 }  
 };**  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState)  
 {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_handler\_thread*);  
  
 //创建后台线程  
 initBackThread();  
  
 mTvServiceInfo = (TextView) findViewById(R.id.*handler\_thread\_text*);  
  
 }  
  
 @Override  
 protected void onResume()  
 {  
 super.onResume();  
 //主线程控制给子线程发送消息 开始互相交互发送消息  
  **Log.i("main", "onResume: 主线程控制让与子线程绑定的handler给子线程自己发送了一条消息 开始互相交互发送消息");  
 handler.sendEmptyMessage(MSG\_UPDATE\_INFO);** }  
  
 @Override  
 protected void onPause()  
 {  
 super.onPause();  
 //停止查询  
 **Log.i("main", "onPause: 主线程控制让与子线程绑定的handler给子线程 停止互相发送消息");  
 handler.removeMessages(MSG\_UPDATE\_INFO);**  
 }  
  
 private void initBackThread()  
 {  
 //HandlerThread类对象实例，初始给线程一个名字  
  **handlerThread = new HandlerThread("check-message-coming");  
 //启动线程，执行其run方法  
 handlerThread.start();  
 //绑定一个handler对象到HandlerThread创建的线程  
 handler = new Handler(handlerThread.getLooper())  
 {  
 //该方法在HandlerThread创建的线程中执行  
 @Override  
 public void handleMessage(Message msg)  
 {  
 checkForUpdate();  
 UIHandler.sendEmptyMessage(1);  
 }  
 };**  
  
 }  
  
 */\*\*  
 \* 模拟从服务器解析数据  
 \*/* private void checkForUpdate()  
 {  
 try  
 {  
 //模拟耗时  
 Thread.*sleep*(1000);  
 //主线程给自己发送消息  
 **Log.i("main", "handleMessage: 子线程控制让主线程给自己发送消息更新UI thread name is "+Thread.currentThread().getName());  
 UIHandler.post(new Runnable()  
 {  
 @Override  
 public void run()  
 {  
 String result = "实时更新中，当前大盘指数：<font color='red'>%d</font>";  
 result = String.format(result, (int) (Math.random() \* 3000 + 1000));  
 mTvServiceInfo.setText(Html.fromHtml(result));  
 }  
 });** } catch (InterruptedException e)  
 {  
 e.printStackTrace();  
 }  
  
 }  
  
 @Override  
 protected void onDestroy()  
 {  
 super.onDestroy();  
 //释放资源  
 Log.*i*("main", "onDestroy: 子线程关闭");  
 **handlerThread.quit();** }  
}