Demo48-Handler的最基本使用

1. 开启子线程new Thread(new Runnable(){ })来执行耗时任务----》可以避免阻塞主线程---》子线程执行完得到结果数据----》返回到主线程更新UI，避免整个应用程序奔溃---》所以子线程中需要载体返回数据到主线程更新UI，两种方式：
2. Handler
3. runOnUiThread( new Runnable(){ })
4. Handler
5. 在子线程中构造Message对象，赋值存储数据,handler.sendMessage(me)
6. 主线程构造Handler匿名类对象，重写handleMessage()方法来处理message对象携带值，进行UI更新操作。

activity\_main.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.constraint.ConstraintLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**Button  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:id="@+id/btn"  
 android:text="handler发送message"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"** />  
  
 <**TextView  
 android:id="@+id/tv\_cpntent"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="How are you ?"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />  
  
</**android.support.constraint.ConstraintLayout**>

MainActivity.java

**package** com.coolweather.handlerdemo;  
  
**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.Button;  
**import** android.widget.TextView;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 **public static final int *TAG***=1;  
 **private** Button **mButton**;  
 **private** TextView **mTextView**;  
  
 **private** Handler **mHandler**=**new** Handler(){  
  
 @Override  
 **public void** handleMessage(Message msg) {  
 **switch** (msg.**what**){  
  
 **case *TAG***:  
 **mTextView**.setText(**"I am fine!Thank U!"**);  
 **break**;  
 **default**:**break**;  
 }  
 }  
 };  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **mTextView**=(TextView)findViewById(R.id.***tv\_cpntent***);  
 **mButton**=(Button)findViewById(R.id.***btn***);  
  
  
 **mButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 Message message=**new** Message();  
 message.**what**=***TAG***;  
 **mHandler**.sendMessage(message);  
 }  
 });  
 }  
}

备注：

1. message对象可以携带基本数据或对象，从子线程返回到主线程。
2. handler对象任务有两个，一个是在子线程发送message，另外一个是重写handleMessage（）方法更新UI
3. MessageQueue是存放message队列
4. Looper是从MessageQueue取message对象分发会handle对象的handleMessage（）方法更新UI
5. runOnUiThread()则是已经封装成了一个接口，可以直接使用。

**异步消息处理机制**

