**Android Thread及其对UI的更新：**

**使用原因：**是在操作一些耗时操作时，比如I/O读写的大文件读写，发起网络请求，数据库操作以及网络下载需要很长时间，为了不阻塞用户界面，出现ANR的响应提示窗口，这个时候我们可以考虑使用Thread线程来解决。

**用法：**

1. 新建一个类继承自Thread，然后重新父类的run（）方法，并在里面编写耗时操作就可以了。

对应的调用方法：new出该类的实例，并调用它的start（）方法。

run（）方法：This method is called when a thread is started that has been created with a class which implements Runnable.（当一个线程开始时这个方法会被调用，它是由一个实现Runable（）的类创建的）

1. 继承的方式耦合程度高，选择实现Runable（）接口的方式，既新建一个类实现Runable（）接口,后重新父类的run（）方法.

对应的调用方法：new出Treat（）的构造函数，该构造函数接收一个Runable参数，并调用它的start（）方法。

1. 如果不想再定义一个类去实现Runable接口，使用匿名类的方式

形式：

01.new Thread(new Runable(){

02.

03. @Override

04. Public void run(){

05. //处理具体的逻辑

06. }

07.}).start();

08.

**子线程对UI的更新**

**主线程：**Service、Activity以及Broadcast均是一个主线程处理，这里我们可以理解为UI线程

**注意：**对于Android平台来说UI控件都没有设计成为线程安全类型，所以更新UI元素，必须在主线程中，所以需要引入一些同步的机制来使其刷新（异步消息处理机制）。

**第一种：异步消息处理机制：**

链接地址：<http://www.cnblogs.com/plokmju/p/android_handler.html>

**四个部分组成：**Message，Handler，MessageQueue（），Looper

**Message：**

作用：定义一个包含描述和任意数据对象的消息，可以用来发送给Handler(处理程序的)，在不同线程之间交换数据

对于Message对象，一般并不推荐直接使用它的构造方法得到，而是建议通过使用Message.obtain()这个静态的方法或者Handler.obtainMessage()获取。Message.obtain()会从消息池中获取一个Message对象，如果消息池中是空的，才会使用构造方法实例化一个新Message，这样有利于消息资源的利用。

**Handler：**用于发送和处理消息或者Runnable对象，

有两大体系：**Post：**Post允许把一个Runnable对象入队到消息队列中。

它的方法有：post(Runnable)、postAtTime(Runnable,long)、postDelayed(Runnable,long)。

**sendMessage：**sendMessage允许把一个包含消息数据的Message对象压入到消息队列中。

它的方法有：sendEmptyMessage(int)、sendMessage(Message)、sendMessageAtTime(Message,long)、sendMessageDelayed(Message,long)。

对于Android中Handler可以传递一些内容，通过Bundle对象可以封装String、Integer以及Blob二进制对象，我们通过在线程中使用Handler对象的sendEmptyMessage或sendMessage方法来传递一个Bundle对象到Handler处理器。对于Handler类提供了重写方法handleMessage(Message msg) 来判断，通过msg.what来区分每条信息。将Bundle解包来实现Handler类更新UI线程中的内容实现控件的刷新操作。

**MessageQueue：**用于存放所有通过Handler发送的消息，每个线程只有一个MessageQueue对象

**Looper：**每个线程中的MessageQueue管家，

Android中每一个Thread都跟着一个Looper，Looper可以帮助Thread维护一个消息队列，同时每一个线程都有一个整数优先级，影响到操作系统的线程是如何安排的。一个新线程继承父的优先级。线程的优先级可以通过setpriority（int）方法设置。

**原理与流程：**

1. 首先在主线程当中创建一个Handler对象，并重写handleMessage（）方法。
2. 当子线程中需要进行UI操作时，就创建一个Message对象，并通过Handler的将这条消息发送出去。
3. 这条消息就被添加到MessageQueue的队列中等待被处理。
4. Looper会一直尝试从MessageQueue中取出待处理的消息，分发到Hander的handleMessage（）方法中。

由于Handler是在主线程中创建的，所以handleMessage（）方法中的代码也会在主线程中运行，所以可以进行相关的UI操作了。

**第二种方法：AsyncTask（封装好的异步消息处理机制）**

**AsyncTask的使用技巧：**在都I你Background（）方法中执行具体的耗时任务，在onProgressUpdate（）方法中进行UI操作，在onPostExecute（）方法中执行一些收尾工作。

**用法：**AsyncTask是一个抽象类，需要创建一个子类去继承它

三个参数用途：

Params:执行AsyncTask所需要的参数（泛型）-----Void

Progress：显示当前进度（泛型）-----integer

Result：对执行结果返回（泛型）------boolean

需要重写的方法：

doInBackground（Params.....）：该方法的所有代码都会在子线程中运行【三个参数都在这里完成：params一开始传进来的参数，代码运行完后通过return返回任务执行结果，由于子线程不能更新UI元素，调用publishProgress（Progress...）方法进行UI操作】-----------只有它是在子线程中

------------------------------------------------------------------------------------------------------

onPreExecute（）---------------初始化操作

onProgressUpdate（Progress...）:给publishProgress（）方法调用的。

onPostExecute（Result）：对return的结果进行相应的操作

**第三种方法：API-runOnUiThread（）方法**

public final void runOnUiThread(Runnable action) {

    if (Thread.currentThread() != mUiThread) {

        mHandler.post(action);

    } else {

        action.run();

    }

}

只是运用了一個 member 來判断是否为UI thread。假设不是，还是要依靠 Handler 来完成接下來的工作。