**使用原因**

**显示的刷新频率是 60Hz，也就是一秒钟刷新 60 次，每 16.67 毫秒刷新一次，为了不丢帧，那么主线程处理代码最好不要超过 16 毫秒,所以一些联网等耗时操作要放在子线程中进行,** **当子线程处理完数据后，为了防止 UI 处理逻辑的混乱，Android 只允许主线程修改 UI，就需要 Handler在子线程和主线程之间通讯,及时刷新ui界面**

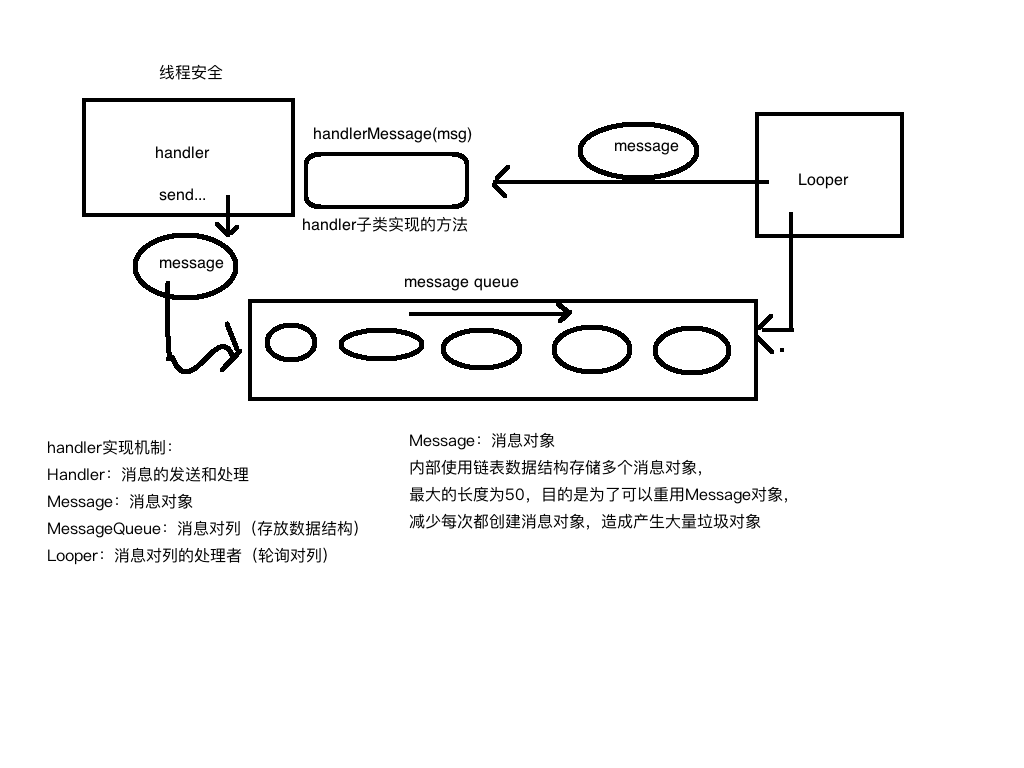
一般用法, Handler 声明在 Activity 中，然后重写 Handler 中的 handleMessage 方法,当子线程调用handler.sendMessage()发送消息后, handleMessage 方法就会在主线程中执行,可以操作UI线程中的view等

1、**Handler本身**：负责消息的发送和处理   
2、**Message**：消息对象   
3、**MessageQueue**：消息队列（用于存放消息对象的[**数据结构**](http://lib.csdn.net/base/datastructure)）   
4、**Looper**：消息队列的处理者（用于轮询消息队列的消息对象，取出后回调handler的dispatchMessage进行消息的分发，dispatchMessage方法会**回调handleMessage**方法把消息传入，由Handler的实现类来处理发送消息接受消息过程

主线程中 Android 默认已经调用了 Looper.preper()方法，调用该方法的目的是在 Looper 中创建MessageQueue 成员变量并把 Looper 对象绑定到当前线程中。当调用 Handler 的 sendMessage（对象）方法的时候就将 Message 对象添加到了 Looper 创建的 MessageQueue 队列中，同时给 Message 指定了 target 对象，其实这个 target 对象就是 Handler 对象。主线程默认执行了 Looper.looper（）方法，该方法从 Looper 的成员变量MessageQueue 中取出 Message，然后调用 Message 的 target 对象的 handleMessage()方法。这样就完成了整个

消息机制

**处理过程**：   
从handler中获取一个消息对象，把数据封装到消息对象中，通过handler的send…方法把消息push到MessageQueue队列中。   
Looper对象会轮询MessageQueue队列，把消息对象取出。   
通过dispatchMessage分发给Handler，再回调用Handler实现的handleMessage方法处理消息。

流程图：   


Handler的实现中适及以下对象：   
1、**Handler本身**：负责消息的发送和处理   
2、**Message**：消息对象   
3、**MessageQueue**：消息队列（用于存放消息对象的[**数据结构**](http://lib.csdn.net/base/datastructure)）   
4、**Looper**：消息队列的处理者（用于轮询消息队列的消息对象，取出后回调handler的dispatchMessage进行消息的分发，dispatchMessage方法会**回调handleMessage**方法把消息传入，由Handler的实现类来处理）

**Message对象**的内部实现是**链表**，最大长度是**50**，用于缓存消息对象，达到重复利用消息对象的目的，以减少消息对象的创建，所以通常我们要使用**obtainMessage**方法来获取消息对象

**安全**：Handler的消息处理机制是**线程安全**的

**关系**:创建Handler时会创建Looper，Looper对象的创建又创建了MessageQueue