使用Handler机制，也就是通过使用Handler，Looper，MessageQueue，和Message这几个类协调来完成。我们先来看使用Handler机制完成线程间通信的原理，然后再详细介绍这几个类；先看一下这几个类的作用：

* Handler，发送和处理Message对象和Runnable对象；
* Looper，用于从MessageQueue中取出消息(Message)对象，并发送给Handler处理；
* MessageQueue，消息队列，用于存放通过Handler发布的消息；
* Message，消息的类型，里面包含几个实例对象：
* what，用户定义的int型消息代码，用来描述消息；
* obj，随消息发送的用户用户指定对象；
* target，处理消息的Handler;

一个Handler对象仅与一个Looper相关联，一个Message也仅与一个目标Handler对象相关联，一个Looper对象拥有一个MessageQueue。但多个不同的Handler对象可以与同一个对象相关联，也就是说多个Handler可以共享一个MessageQueue，从而达到消息共享的目的，这也是Android通过Handler机制实现多线程间通信的核心原理；

上面说到Handler对象仅与一个Looper相关联，那么这个关联是什么时候实现的呢？答案是：Handler对象创建的时候。UI主线程在创建的时候就会拥有一个handler对象和一个Looper对象（工作线程需要自己调用Looper.prepare()来创建一个Looper对象），然后任何在UI主线程中创建的Handler对象默认都会与UI主线程的Looper对象相关联（Handler对象创建的时候，会与这个线程的Looper对象相关联）。进而我们就可以把在UI主线程中创建的Handler对象传递给（依赖注入或引用形式）工作线程，那么在工作线程中用这个Handler对象处理的消息就是在UI主线程的MessageQueue中处理的，从而达到线程间通信的目的。

了解了使用Handler机制来实现Android线程间异步通信的原理，下面我们再来详细了解下这四个核心类；

**2.2.1 Handler**

Handler，继承自Object类，用来发送和处理Message对象或Runnable对象；Handler在创建时会与当前所在的线程的Looper对象相关联(如果当前线程的Looper为空或不存在，则会抛出异常，此时需要在线程中主动调用Looper.prepare()来创建一个Looper对象)。使用Handler的主要作用就是在后面的过程中发送和处理Message对象和让其他的线程完成某一个动作（如在工作线程中通过Handler对象发送一个Message对象，让UI线程进行UI的更新，然后UI线程就会在MessageQueue中得到这个Message对象（取出Message对象是由其相关联的Looper对象完成的），并作出相应的响应）。

Handler用post体系来完成发送Runnable对象的工作，用sendMessage体系 来完成发送Message对象的工作；

 post体系,允许把一个Runnable对象发送到消息队列中，它的方法有：post(Runnable),postDelayed(Runnable,long),postAtTime(Runnable,long)；

 sendMessage体系,把Message对象发送给消息队列，它的方法有：sendEmptyMessage(int),sendMessage(Message),sendMessageDelayed(Message,long),sendMessageAtTime(Message,long);

如果Handler是通过post体系将Runnable对象发送到MessageQueue队列中，则这个Runnable对象的run()方法是运行在Handler对象创建时所在线程；  
如果Handler是通过sendMessage体系将Message发送到MessageQueue中，则需要重写handleMessage()方法来获取工作线程传递过来的Message对象，handleMessage()方法是工作在Handler对象建立时所在的线程的（一般我们会在UI线程中建立Handler对象，然后传递给子线程，此时这个Handler对象的handleMessage()方法就是运行在UI主线程中的）；

**2.2.2 Message**

Message用来定义一个包含任意数据的消息对象，这个消息对象是可以被发送给Handler处理的。我们最好通过Message.obtain()和Handler.obtatinMessage()来得到一个Message对象（通过这两个方法得到的对象是从对象回收池中得到，也就是说是复用已经处理完的Message对象，而不是重新生成一个新对象），如果通过Message的构造方法得到一个Message对象，则这个Message对象是重新生成的（不建议使用这种方法）。

Message对象用来封装需要传递的消息，Message的数据结构为：

Message{

int arg1;//如果我们只需要存储一些简单的Integer数据，则可通过设置这个属性来传递int agr2;//使用同arg1

Object obj; //设置需要发送给接收方的对象,这个对象需要实现序列化接口int what; //描述这个消息的标识；//设置与这个消息对应的任意数据，这个数据是用Bundle封装的；voidsetData(Bundle data);

Bundle getData(); 得到与这个消息对应的数据信息;

//省略了方法和可选的属性

......

如果需要通过Message对象传递一些比较复杂的数据，则需要使用将数据封装成Bundle对象，然后通过setData(Bundle)方法来传递，用getData()来得到与这个消息对应的数据(这方法与设置Message的Object 属性作用相同);

**2.2.3 MessageQueue**

MessageQueue保存由Looper调度的消息列表，消息通过与Looper相关联的Handler对象添加进MessageQueue。

**2.2.4 Looper**

Looper为线程运行一个消息的循环队列，主要就是为了完成MessageQueue与Handler交互的功能；需要注意的是线程默认并不会给我们提供一个一个Looper实例来管理消息队列，我们需要在线程中主动调用Looper.prepare()方法来实例化一个Looper对象，用于管理消息队列；Looper对象会不断去判断MessageQueue是否为空，如果不空，则将Message取出给相应的Handler进行处理；如果MessageQueue为空，则Looper对象会进行阻塞状态，直到有新的消息进入MessageQueue；