**05\_消息机制与异步任务**

🖊 人不光是靠他生来就拥有一切，而是靠他从学习中所得到的一切来造就自己。 —— 歌德

**相关课程资料：**

鼠标放置在正文左侧，点击“**+**”工具栏中的“**高亮块**”来插入高亮块。

|  |
| --- |
| 科目：Android  日期：03/26  主题：品牌定位  课前阅读： @ 插入相关文档 |

**Runnable对象的run（）什么时候在分线程执行？**

将runnable传给Thread的构造方法

|  |
| --- |
| Java  new Thread(new Runnable(){  public void run(){  // 在分线程执行  }  }).start(); |

**消息机制的引入**

1. 在Android中，只有在UIThread中才能直接更新界面
2. 在Android中，长时间的工作（联网）都需要在workerThread中执行
3. 在分线程获得服务器数据后，需要立即到主线程去更新界面显示数据

**如何实现线程间通信呢？**

1. 消息机制：Handler + Thread
2. 异步任务： AsyncTask === Handler + ThreadPool

**Message的使用：线程间通信的数据载体**

1. **对象的创建**
2. Message.obtain（） 有可能是新建的，也可能是复用的消息池中
3. **对象的使用**
4. Int what 标识
5. Int args1 用于保存int数据
6. Int args2 用于保存int数据
7. Object obj 用于保存对象数据
8. Handler target 保存处理消息的handler
9. Long when 消息应该被处理的时间
10. Message next 有了次属性Message就可以形成队列
11. Static Message pool 消息池，缓存用过的消息用于复用
12. Runnable callBack 用于处理当前消息的回调器

**Handler的使用：发送 / 处理 / 移除消息**

1. **对象的创建：new Handler()**
2. **对象的使用：**
3. **发送消息**
4. 发即时消息：sendMessage（Message message）
5. 发延迟消息：sendMessageDelayed（Message message，long dalayTime）
6. 发即时空消息：sendEmptyMessage（int what）
7. 发送延迟空消息：sendEmptyMessageDelayed（int what，long delayTime）
8. **处理消息**
9. Void handleMessage（Message msg）回调方法
10. **移除消息**
11. removeMessage（int what）移除未处理的指定what的message
12. removeCallbacksAndMessages（null）：移除所有未处理的消息

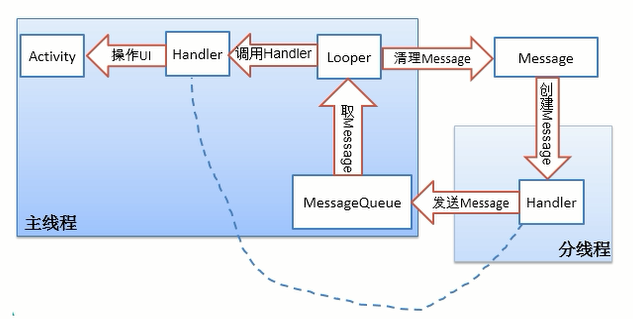
**MessageQueue：消息队列**

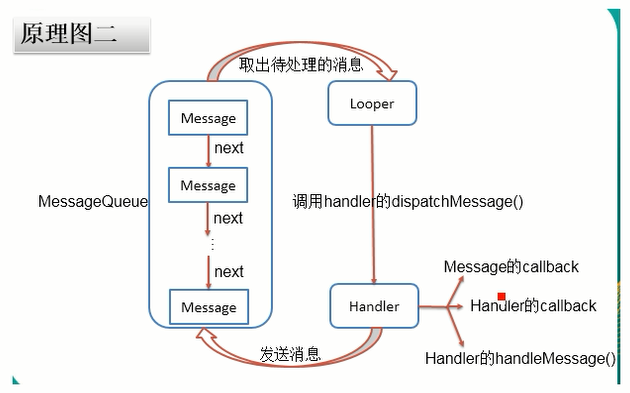
1. 用来存放通过Handler发送的消息
2. 它是一个按Message的when排序的优先级队列

**Looper（钩子）:循环器**

1. 负责循环取出Message Queue里面的当前需要处理的Message
2. 交给对应的Handler进行处理
3. 处完后，将Message缓存到消息池中，以备复用

**消息机制的原理**



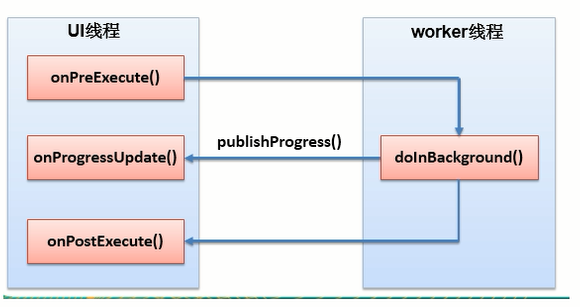


**Handler用于做什么工作？**

1. 线程间通信（分线程间切换到主线程运行）
2. 延迟的工作
3. 循环定时工作

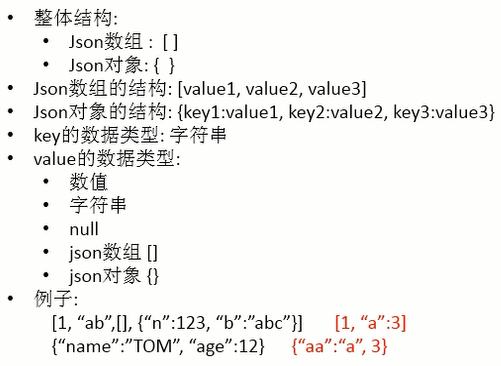
**异步任务AsyncTask**

1. **什么是异步任务？**
2. 逻辑上：以多线程的方式完成的功能需求
3. API上：指AsyncTask类
4. **AsyncTask的理解**
5. 在没有AsyncTask之前，我们用Handler + Thread就可以实现异步任务的功能需求
6. AsyncTask是对Handler和Thread的封装，使用它更编码更简洁，更高效
7. AsyncTask封装了ThreadPool，比直接使用Thread效率要高



**JSON数据格式和解析**

1. **JSON是什么？**
2. 有特定格式的字符串
3. 用于客户端与服务器的数据交互的一种数据格式
4. 与XML比较的优势？
5. 小、简洁
6. 省流量、响应快
7. **JSON数据格式**



1. **如何解析JSON？**

