HenCoder Plus 第 22 课 讲义

线程间通信的本质和原理，以及 Android 中的多线程

线程间交互

- 一个线程启动别的线程：new Thread().start()、Executor.execute() 等
- 一个线程终结另一个线程
  - Thread.stop()
  - Thread.interrupt()：温和式终结：不立即、不强制
    - interrupted() 和 isInterrupted()：检查（和重置）中断状态
    - InterruptedException：如果在线程「等待」时中断，或者在中断状态「等待」，直接结束等待过程（因为等待过程什么也不会做，而 interrupt() 的目的是让线程做完收尾工作后尽快终结，所以要跳过等待过程）
- Object.wait() 和 Object.notify() / notifyAll()
  - 在未达到目标时 wait()
  - 用 while 循环检查
  - 设置完成后 notifyAll()
  - wait() 和 notify() / notifyAll() 都需要放在同步代码块里
- Thread.join()：让另一个线程插在自己前面
- Thread.yield()：暂时让出自己的时间片给同优先级的线程

Android 的 Handler 机制

- 本质：在某个指定的运行中的线程上执行代码
- 思路：在接受任务的线程上执行循环判断
- 基本实现：
  - Thread 里 while 循环检查
  - 加上 Looper（优势在于自定义 Thread 的代码可以少写很多）：
  - 再加上 Handler（优势在于功能分拆，而且可以有多个 Handler）
- Java 的 Handler 机制：
  - HandlerThread：具体的线程
  - Looper：负责循环、条件判断和任务执行
  - Handler：负责任务的定制和线程间传递
- AsyncTask：
  - AsyncTask 的内存泄露

- 众所周知的原因：AsyncTask 持有外部 Activity 的引用
- 没提到的原因：执行中的线程不会被系统回收
- Java 回收策略：没有被 GC Root 直接或间接持有引用的对象，会被回收

  GC Root：

  1. 运行中的线程
  2. 静态对象
  3. 来自 native code 中的引用
- 所以：
  - AsyncTask 的内存泄露，其他类型的线程方案（Thread、Executor、HandlerThread）一样都有，所以不要忽略它们，或者认为 AsyncTask 比别的方案更危险。并没有。
  - 就算是使用 AsyncTask，只要任务的时间不长（例如 10 秒之内），那就完全没必要做防止内存泄露的处理。

Service 和 IntentService

- Service：后台任务的活动空间。适用场景：音乐播放器等。
- IntentService：执行单个任务后自动关闭的 Service

Executor、AsyncTask、HandlerThead、IntentService 的选择

原则：哪个简单用哪个

- 能用 Executor 就用 Executor
- 需要用到「后台线程推送任务到 UI 线程」时，再考虑 AsyncTask 或者 Handler
- HandlerThread 的使用场景：原本它设计的使用场景是「在已经运行的指定线程上执行代码」，但现实开发中，除了主线程之外，几乎没有这种需求，因为 HandlerThread 和 Executor 相比在实际应用中并没什么优势，反而用起来会麻烦一点。不过，这二者喜欢用谁就用谁吧。
- IntentService：首先，它是一个 Service；另外，它在处理线程本身，没有比 Executor 有任何优势

关于 Executor 和 HandlerThread 的关闭

如果在界面组件里创建 Executor 或者 HandlerThread，记得要在关闭的时候（例如 `Activity.onDestroy()`）关闭 Executor 和 HandlerThread。

```
@Override
protected void onDestroy() {
    super.onDestroy();
    executor.shutdown();
}
```

```
@Override
protected void onDestroy() {
    super.onDestroy();
    handlerThread.quit(); // 这个其实就是停止 Looper 的循环
}
```

问题和建议?

课上技术相关的问题，都可以在学员群里和大家讨论，我一旦有时间也都会来解答。如果我没来就 @ 我一下吧！

具体技术之外的问题和建议，都可以找丢物线（微信：diuwuxian），丢丢会为你解答技术以外的一切。

更多内容：

- 网站：https://hencoder.com
- 微信公众号：HenCoder

HenCoder

给高级 Android 工程师的进阶手册

微信公众号：HenCoder
微博：扔物线
知乎专栏：HenCoder
稀土掘金：扔物线
http://hencoder.com