

Re: Do I need Threads for this?

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2006-12/msg00915.html>

- *From:* "Oliver Wong" <owong@xxxxxxxxxxxxxxxx>
 - *Date:* Tue, 5 Dec 2006 11:00:44 -0500
-

<nospam@xxxxxxxxxxxxxxxx> wrote in message
news:loWdnZQDa8Q3DOjYRVnyuQ@xxxxxxxx

Yes, you need threads, but it's not that hard. You make a class that implements Runnable or extends Thread. Then you can do all ten files at once, if you choose to!

Hmm, seems a bit awkward, but then Java seems that way sometimes...

Methodically, I can't understand how by splitting the work into two threads its going to update the label at the time I need it to – won't it just split into two and one process copy over the files in one thread all at once and update the labels all at once in the other, meaning that there won't be any real live interaction between the two? I guess I'm really saying that I don't understand how the thread processes interact with each other...

Presumably the thread which is doing the file-copying will update some region in memory, recording it's progress. E.g. "Okay, I'm done with file #4... now I'm done with #5... #6...", periodically releasing the CPU (actually, once it tells the HD to start copying, it can immediately release the CPU, because it has to wait for the HD to actually perform the reads and the writes, only to be notified when the copy has finished, to schedule more copies, and go back to sleep).

Meanwhile, the main thread will be scanning this region of memory and see "Okay, so that thread is done with file #6. That means I need to change the label to say 'Files copied: 6', and then release the CPU so that the implicit thread that AWT/Swing creates has a chance to actually draw the changes I've made to screen."

– Oliver

.