**金立线程池**

**使用线程池的好处:**

1. 减少在创建和销毁线程上所花的时间以及系统资源的开销
2. 如不使用线程池，有可能造成系统创建大量线程而导致消耗完系统内存
3. 统一管理

**在什么情况下使用线程池？**

1. 单个任务处理的时间比较短
2. 将需处理的任务的数量大

**我们的目标：**

1. 可以多任务同时返回
2. 支持任务的拆分和结果合并---------
3. 给每个任务取一个名字，方便问题的跟踪
4. 可以定义线程的执行优先级、支持任务的插队、排序
5. 可以调用Runnable和Callable
6. 可以确定任务的执行状态-------
7. 可以取消任务
8. 可以关闭线程池
9. 提供默认的线程池

**线程池角色说明：**

//Task、ResultTask：调用方提供的一个执行任务，定义了是否需要插队以及插队的优先级，//提供上层回调的方法。

//NameRunnable：内部实现设置任务所在线程名字，这样线程运行出问题时就能知道当前运//行的是哪个任务。

//Future：获得任务的执行返回值。支持取消任务、延时获取消息

//Disposable：处理任务缓存池的任务，判断是否被处理或移除

//Worker：装载一个Task，并返回Disposable

//TaskCachePool：线程池被阻塞时，任务临时存放地址，该任务缓存池支持插队、移除。

//SchedulerFactory：初始化不同策略的线程池，比如I/O操作、纯CPU计算操作、多任务操//作、任务拆分操作等。也支持调用方自定义线程池，统一管理默认的线程池。

//Scheduler：提供创建线程池、关闭线程池、执行worker、返回执行结果等，并且提供正在//运行的线程和等待执行的任务，用于动态调整核心线程数，不过动态调整动作应该交由调//用方，该类是抽象类，需子类提供具体实现。

//Generater：线程池模块的入口类，提供封装一个或多个Task、调用线程池、延时处理Task、//执行Task等操作，每个操作都会生成一个Generater的子类。执行顺序为根据方法调用的//顺序从后向前执行。

//ForkJoinExcutor：处理单任务进行分解执行并最终合并结果。

//ExecutorBatchService：处理多任务同时返回结果。

//ThreadFactory：真正创建线程的工厂，需子类实现特殊逻辑，比如设置线程的优先级等