ExecutorService接口源码阅读

# 阅读

package java.util.concurrent;  
  
import java.util.List;  
import java.util.Collection;  
  
public interface ExecutorService extends Executor {  
  
 */\*\*  
 \* 启动有序关闭，继续执行完之前提交的任务，但是不再接收新任务。  
 \*/* void shutdown();  
  
 */\*\*  
 \* 尝试停止目前正在执行的任务，对于等待队列中的待执行任务不再处理。本方法只能尽最大努力  
 \* 停止正在执行的任务，无法保证任务一定停止执行，例如：任何无法响应中断的任务都可能永远  
 \* 不会终止。  
 \*  
 \** ***@return*** *从未开始执行的任务列表  
 \*/* List<Runnable> shutdownNow();  
  
 boolean isShutdown();  
  
 */\*\*  
 \* 如果所有任务在ExecutorService启动关闭程序后都被成功关闭则放回true，否则返回false。  
 \* 需要注意，除非先调用shutdown()或者shutdownNow()，否则isTerminated()永远返回false。  
 \*  
 \** ***@return*** *\*/* boolean isTerminated();  
  
 */\*\*  
 \* 阻塞，直到所有调用shutdown()时正在执行的任务都执行完成，或者达到超时时间，或者发生中断，  
 \* 任意一种条件满足都会终止阻塞。  
 \*  
 \** ***@param*** *timeout - 最大等待时间  
 \** ***@param*** *unit - timeout参数的时间单位  
 \** ***@return*** *达到超时时间前所有任务执行完成，返回true，否则返回false。  
 \** ***@throws*** *InterruptedException  
 \*/* boolean awaitTermination(long timeout, TimeUnit unit)  
 throws InterruptedException;  
  
 */\*\*  
 \* Submits a value-returning task for execution  
 \*  
 \** ***@param*** *task - the task to submit  
 \** ***@param*** <*T*> *- the type of the task's result  
 \** ***@return*** *Future对象供客户端查看任务执行的情况，以及控制任务（Future类中的get方法返回任务执行）  
 \*/* <T> Future<T> submit(Callable<T> task);  
  
 */\*\*  
 \* Submits a Runnable task for execution  
 \*  
 \** ***@param*** *task - the task to submit  
 \** ***@param*** *result - the result to return  
 \** ***@param*** <*T*> *- the type of the result  
 \** ***@return*** *- a Future representing pending completion of the task  
 \*/* <T> Future<T> submit(Runnable task, T result);  
  
 */\*\*  
 \* Submits a Runnable task for execution and returns a Future representing that task.  
 \* The Future's get method will return null upon successful completion.  
 \*/* Future<?> submit(Runnable task);  
  
 */\*\*  
 \* Executes the given tasks, returning a list of Futures holding their status and results when all complete.  
 \* Future.isDone() is true for each element of the returned list.  
 \* Note that a completed task could have terminated either normally or by throwing an exception. The results  
 \* of this method are undefined if the given collection is modified while this operation is in progress.  
 \* (一个已完成的任务可能已正常终止，也可能引发异常。如果在进行此操作时修改了给定的集合，则此方法的结果不确定。)  
 \*  
 \** ***@param*** *tasks - the collection of tasks  
 \** ***@param*** <*T*> *- the type of the values returned from the tasks  
 \** ***@return*** *a list of Futures representing the tasks, in the same sequential order as produced by the iterator  
 \* for the given task list, each of which has completed  
 \** ***@throws*** *InterruptedException - if interrupted while waiting, in which case unfinished tasks are cancelled  
 \*/* <T> List<Future<T>> invokeAll(Collection<? extends Callable<T>> tasks)  
 throws InterruptedException;  
  
 */\*\*  
 \* Executes the given tasks, returning a list of Futures holding their status and results when all complete or  
 \* the timeout expires, whichever happens first.Future.isDone() is true for each element of the returned list.  
 \* Upon return, tasks that have not completed are cancelled. Note that a completed task could have terminated  
 \* either normally or by throwing an exception. The results of this method are undefined if the given collection  
 \* is modified while this operation is in progress.  
 \*  
 \** ***@param*** *tasks  
 \** ***@param*** *timeout - the maximum time to wait  
 \** ***@param*** *unit - the time unit of the timeout argument  
 \** ***@param*** <*T*>  
 *\** ***@return*** *If the operation did not time out, each task will have completed. If it did time out,  
 \* some of these tasks will not have completed.  
 \** ***@throws*** *InterruptedException  
 \*/* <T> List<Future<T>> invokeAll(Collection<? extends Callable<T>> tasks,  
 long timeout, TimeUnit unit)  
 throws InterruptedException;  
  
 */\*\*  
 \* 执行给定的任务，如果成功，则返回成功完成任务（即不引发异常）的结果。在正常或异常返回时，尚未完成的任务将被取消。  
 \* 如果在进行此操作时修改了给定的集合，则此方法的结果不确定。  
 \*/* <T> T invokeAny(Collection<? extends Callable<T>> tasks)  
 throws InterruptedException, ExecutionException;  
  
 */\*\*  
 \* 执行给定的任务，如果在给定的超时时间过去之前完成任务，则返回成功完成任务的结果（即没有引发异常）。在正常或异常返回时，  
 \* 尚未完成的任务将被取消。如果在进行此操作时修改了给定的集合，则此方法的结果不确定。  
 \*/* <T> T invokeAny(Collection<? extends Callable<T>> tasks,  
 long timeout, TimeUnit unit)  
 throws InterruptedException, ExecutionException, TimeoutException;  
}

# 总结

1. ExecutorService接口拓展自Executor接口，一般很少直接使用Executor接口，大多数情况下使用的是ExecutorService接口。
2. 客户端将任务执行的工作外包给ExecutorService接口，实现任务创建和任务执行的解耦，符合接口隔离原则。
3. ExecutorService中的方法可以分为3类：

1)任务提交

2)执行器关闭

3)状态查询