

# Executor接口

## void execute(Runnable command)

异步执行command，避免重复的new Thread消耗资源，更便于统一管理。

# ExecotorService接口

## void shutdown()

不再接受新任务，等待所有任务结束后关闭ExectorService。

## List<Runnable> shutdownNow()

直接关闭ExectorService，正在执行的线程interrupt并把列表返回。

## boolean isShutdown()

判断ExectorService是否为关闭状态。

## boolean isTerminated()

判断ExectorService是否是在所有任务都完成后关闭的。

## boolean awaitTermination(long timeout, TimeUnit unit) throws InterruptedException

等待单位是unit的时候timeout后返回ExectorService是否全部执行完。

## <T> Future<T> submit(Callable<T> task)

提交task任务，得到Future对象，可以得到任务执行的结果。

## <T> Future<T> submit(Runnable task, T result)

提交task任务，成功后调用future.get()将返回result。

## Future<?> submit(Runnable task)

提交task任务，成功后调用future.get()将返回null。

## <T> List<Future<T>> invokeAll(Collection<? extends Callable<T>> tasks) throws InterruptedException

批量提交任务并获得他们的future，Task列表与Future列表一一对应（全部执行完返回）。

<T> List<Future<T>> invokeAll(Collection<? extends Callable<T>> tasks,long timeout, TimeUnit unit)  
 throws InterruptedException

批量提交任务并获得他们的future，并限定处理所有任务的时间。

<T> T invokeAny(Collection<? extends Callable<T>> tasks)  
 throws InterruptedException, ExecutionException

批量提交任务并获得一个已经成功执行的任务的结果(只要成功执行一个就返回)。

<T> T invokeAny(Collection<? extends Callable<T>> tasks,long timeout, TimeUnit unit)  
 throws InterruptedException, ExecutionException, TimeoutException

批量提交任务，并限定处理所有任务的时间。返回一个已经成功执行的任务的结果(只要成功执行一个就返回)。