Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

IES Mare Nostrum

Programación Concurrente

Gestión de múltiples hilos

Introducción

Problema

- La creación de hilos es un proceso relativamente costoso.

Solución

- Interfaz *ExcutorService* ➞ gestión eficiente / reutilización de hilos.

ExecutorService

Crea un *pool* o conjunto de hilos que se reutilizan según se necesiten para ejecutar las diferentes tareas.

`void execute (Runnable command)`

Ejecuta en un hilo la tarea que recibe por parámetro.

`void shutdown()`

Hace que el executor no acepte más llamadas. El executor esperará a que terminen de ejecutarse las tareas que se encuentren en ejecución y las que estén a la espera de ejecutarse. Una vez finalizadas se cierra el executor.

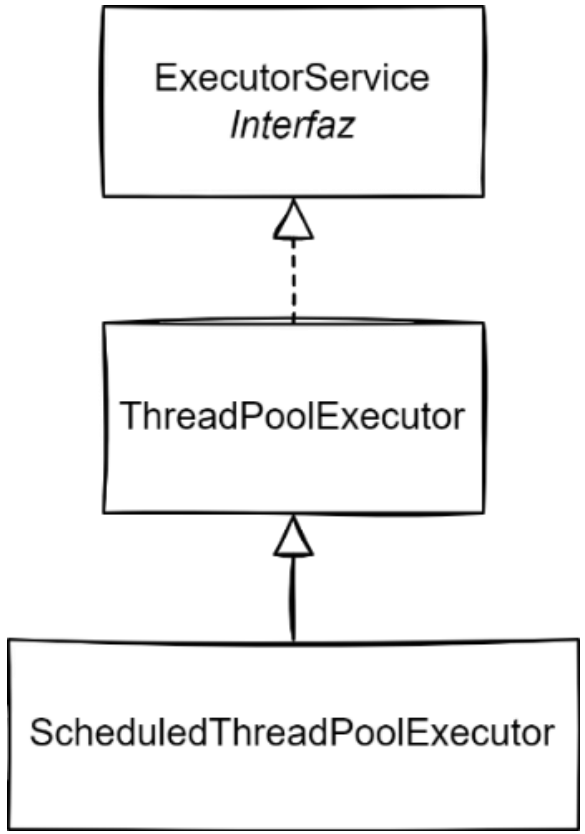
`List<Runnable> shutdownNow()`

Finaliza las tareas en ejecución de forma abrupta. Devuelve una lista con las tareas que estaban a la espera de ejecutarse.

`boolean awaitTermination(long timeout, TimeUnit unit)`

Llamada bloqueante que espera hasta que todas las tareas finalicen su ejecución o hasta que finalice la espera indicada en timeout. Este método se suele llamar tras llamar al método shutdown par esperar la finalización de las tareas.

ExecutorService



ThreadPoolExecutor

Implementa *ExecutorService*.

Constructores con un gran número de parámetros.

```
ThreadPoolExecutor(int corePoolSize, int maximumPoolSize, long keepAliveTime, TimeUnit unit, BlockingQueue<Runnable> workQueue)

ThreadPoolExecutor(int corePoolSize, int maximumPoolSize, long keepAliveTime, TimeUnit unit, BlockingQueue<Runnable> workQueue,
RejectedExecutionHandler handler)

ThreadPoolExecutor(int corePoolSize, int maximumPoolSize, long keepAliveTime, TimeUnit unit, BlockingQueue<Runnable> workQueue,
ThreadFactory threadFactory)

ThreadPoolExecutor(int corePoolSize, int maximumPoolSize, long keepAliveTime, TimeUnit unit, BlockingQueue<Runnable> workQueue,
ThreadFactory threadFactory, RejectedExecutionHandler handler)
```

Métodos que nos interesan:

`public int getPoolSize()`

Devuelve el número de hilos que existen en el pool.

`public int getActiveCount()`

Devuelve el número de hilos que están ejecutando alguna tarea.

ScheduledExecutorService

Hereda de *ThreadPoolExecutor*.

Constructores con un gran número de parámetros.

Métodos que nos interesan:

```
ScheduledFuture<?> schedule(Runnable command, long delay, TimeUnit unit)
```

Lanza una tarea cuando transcurra el tiempo indicado en *delay*. El parámetro *unit* indica la unidad de tiempo utilizada.

```
ScheduledFuture<?> scheduleAtFixedRate(Runnable command, long initialDelay,
                                       long period, TimeUnit unit)
```

Lanza una tarea una vez pasado el tiempo indicado en el parámetro *initialDelay* y la ejecutará repetidamente con la frecuencia indicada en el parámetro *period*.

Executors

Factory que facilita la instanciación de objetos de tipo *ThreadPoolExecutor* y *ScheduledThreadPoolExecutor*.

`static ExecutorService newFixedThreadPool(int nThreads)` — *Ejemplo 23 y 23b*

Crea un *thread pool* que reutiliza un número fijo de hilos *nThreads* con una cola de espera ilimitada.

`static ExecutorService newSingleThreadExecutor()` — *Ejemplo 24*

Crea un *thread pool* de un único hilo con una cola de espera ilimitada.

`static ExecutorService newCachedThreadPool()` — *Ejemplo 25*

Crea un *thread pool* que permite crear tantos hilos nuevos como resulten necesarios, reutilizando los que se hubiesen construido previamente si están disponibles.

Executors

Factory que facilita la instanciación de objetos de tipo *ThreadPoolExecutor* y *ScheduledThreadPoolExecutor*.

`static ScheduledExecutorService newScheduledThreadPool(int corePoolSize)` *Ejemplo 26 y 27*

Crea un *thread pool* de un número fijo de hilos *corePoolSize* hilos que se ejecutarán tras un determinado tiempo de espera o periódicamente.

`static ScheduledExecutorService newSingleThreadScheduledExecutor()`

Crea un *thread pool* que permite programar un único hilo que se ejecutará tras un determinado tiempo de espera o periódicamente