

Tabla de Contenidos



- ¿Qué es ExecutorService?
- Cómo usar ExecutorService
- Métodos principales de ExecutorService
- Obtener resultados con Callable y Future
- Ventajas de ExecutorService
- Consejos al usar ExecutorService
- Conclusión

La programación concurrente es clave en el desarrollo de aplicaciones modernas. En Java, la interfaz `ExecutorService` nos ayuda a manejar múltiples tareas en paralelo sin complicarnos con la creación manual de hilos como antiguamente sucedía al usar `Threads`. En este artículo, veremos qué es `ExecutorService`, cómo usarlo y sus beneficios.

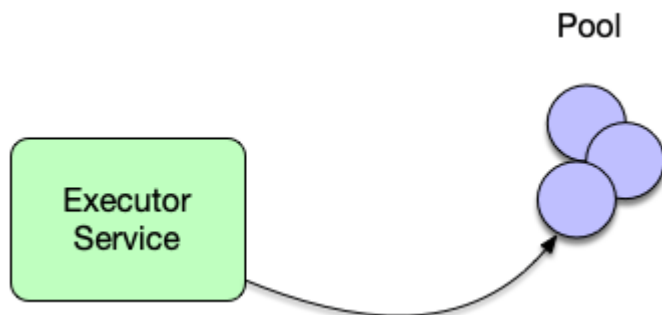
¿Qué es ExecutorService?

`ExecutorService` es una interfaz en Java que forma parte del paquete `java.util.concurrent`. Su función principal es gestionar un grupo de hilos para ejecutar tareas sin que tengamos que crearlos y administrarlos uno por uno.

Cómo usar ExecutorService

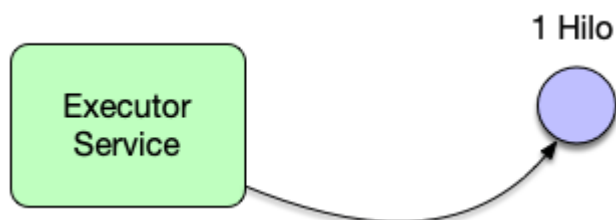
Para crear un `ExecutorService`, lo más fácil es usar los métodos de la clase `Executors`.

El método `newFixedThreadPool(int n)` crea un pool con un número fijo de hilos. Si un hilo está ocupado, las tareas esperan en una cola.



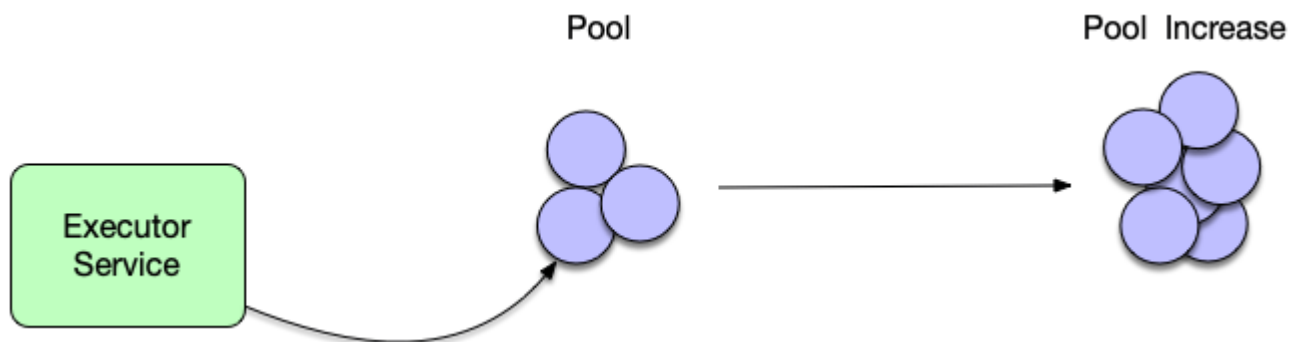
```
ExecutorService executor = Executors.newFixedThreadPool(3);  
for (int i = 0; i < 5; i++) {  
    executor.execute(() -> System.out.println("Ejecutando tarea en: " +  
    Thread.currentThread().getName()));  
}  
executor.shutdown();
```

El método `newSingleThreadExecutor()` crea un único hilo para ejecutar tareas en orden secuencial



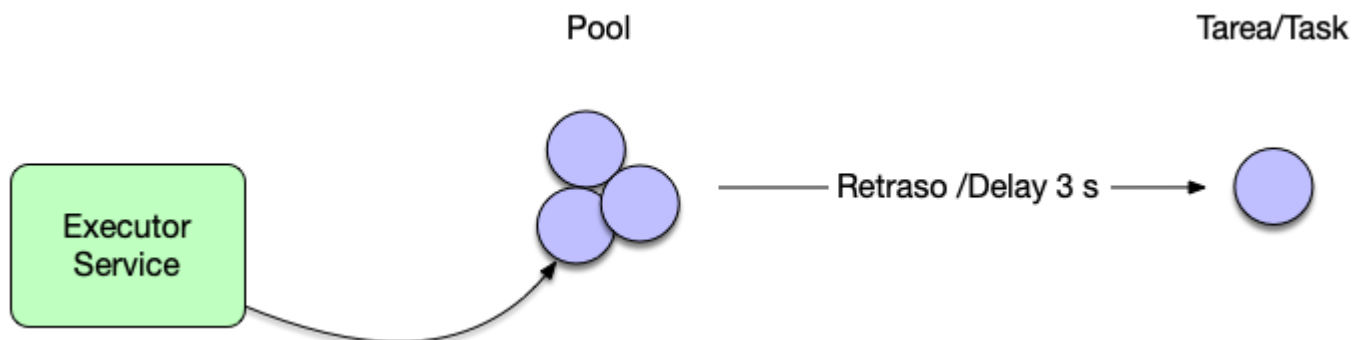
```
ExecutorService executor = Executors.newSingleThreadExecutor();  
executor.execute(() -> System.out.println("Ejecutando tarea en un solo  
hilo"));  
executor.shutdown();
```

El método `newCachedThreadPool()` crea un pool de hilos que se expande dinámicamente según la carga de trabajo.



```
ExecutorService executor = Executors.newCachedThreadPool();
for (int i = 0; i < 5; i++) {
    executor.execute(() -> System.out.println("Tarea ejecutada por: "
+ Thread.currentThread().getName()));
}
executor.shutdown();
```

El método `newScheduledThreadPool(int corePoolSize)` permite ejecutar tareas programadas con un retraso o de manera repetitiva.



```
ScheduledExecutorService executor =
Executors.newScheduledThreadPool(2);
executor.schedule(() -> System.out.println("Tarea ejecutada después de
3 segundos"), 3, TimeUnit.SECONDS);
executor.shutdown();
```

Métodos principales de ExecutorService

Algunos métodos útiles que ofrece ExecutorService:

- `execute(Runnable task)`: Ejecuta una tarea en segundo plano sin esperar un resultado.
- `submit(Callable task)`: Igual que `execute`, pero devuelve un resultado con `Future`.
- `shutdown()`: Detiene el `ExecutorService` después de terminar las tareas en curso.
- `shutdownNow()`: Intenta detener todas las tareas de inmediato.
- `invokeAll(Collection<Callable> tasks)`: Ejecuta varias tareas y devuelve una lista de objetos `Future` con los resultados.
- `invokeAny(Collection<Callable> tasks)`: Ejecuta varias tareas y devuelve el resultado de la primera que termine.

Obtener resultados con Callable y Future

Si necesitamos que una tarea devuelva un resultado, usamos `Callable` en vez de `Runnable`.

```
ExecutorService executor = Executors.newSingleThreadExecutor();
Callable<Integer> task = () -> {
    Thread.sleep(1000);
    return 42;
};
Future<Integer> future = executor.submit(task);
try {
    System.out.println("Resultado: " + future.get());
} catch (InterruptedException | ExecutionException e) {
    e.printStackTrace();
} finally {
    executor.shutdown();
}
```

Ventajas de ExecutorService

Usar ExecutorService en lugar de manejar hilos manualmente tiene muchas ventajas:

- Reutilización de hilos: Se evita crear y destruir hilos constantemente.
- Mejor control: Es más fácil administrar tareas en paralelo.
- Escalabilidad: Se puede ajustar el número de hilos según las necesidades.
- Manejo de errores: Future permite capturar excepciones de tareas asíncronas.

Consejos al usar ExecutorService

- Siempre usa shutdown() o shutdownNow() para liberar recursos.
- Configura bien el tamaño del pool de hilos para evitar sobrecarga.
- Para tareas repetitivas, usa ScheduledThreadPoolExecutor en vez de Timer.

Conclusión

ExecutorService es una excelente herramienta para manejar múltiples tareas en Java de forma sencilla y eficiente. Permite mejorar el rendimiento sin la complejidad de administrar hilos manualmente. Si trabajas con concurrencia en Java, esta es una solución que debes considerar para mejorar la eficiencia y escalabilidad de tus aplicaciones.

- [Java Executor Service y Threading](#)
- [Java Callable Interface y su uso](#)
- [Servlet 3.0 \(II\) Servlets Asincronos](#)
- [Command Pattern en Java y la gestion de tareas](#)
- [Java ResultSet un ejemplo Sencillo con Persona](#)