</> forkjoin

Fernando Duque

Daniel Yuste

Daniel Serrano

MÉTODOS

forkjointask

forkjoinpool

recursivetask

- Fork(): Este método divide la tarea en subtareas más pequeñas y las ejecuta en paralelo. Es el método más importante para ejecutar tareas ForkJoinTask en paralelo.
- Task(): Este método crea una nueva tarea ForkJoinTask. Es útil para crear tareas personalizadas que se pueden ejecutar en un ForkJoinPool.
- Join(): Este método espera a que la tarea se complete. Esto es útil para obtener el resultado de una tarea ForkJoinTask.
- Compute(): Este método es el más importante de la clase ForkJoinTask. Ejecuta la lógica de la tarea. Si la tarea es lo suficientemente pequeña, la ejecuta de manera secuencial. Si no, la divide en subtareas más pequeñas y las ejecuta en paralelo.
- Pool(): Este método obtiene el pool de hilos que se utilizará para ejecutar la tarea. Esto es útil para tareas que requieren un pool de hilos específico.
- Invoke(): Este método ejecuta la tarea de manera secuencial. Es útil para tareas que no se pueden dividir en subtareas más pequeñas, opara tareas que se ejecutan mejor secuencialmente.
- Adapt(): Este método convierte una tarea en otra tarea ForkJoinTask compatible. Esto es útil para tareas que no son tareas ForkJoinTask, pero que se pueden ejecutar en un ForkJoinPool.

VENTAJAS Y DESVENTAJAS

VENTAJAS:

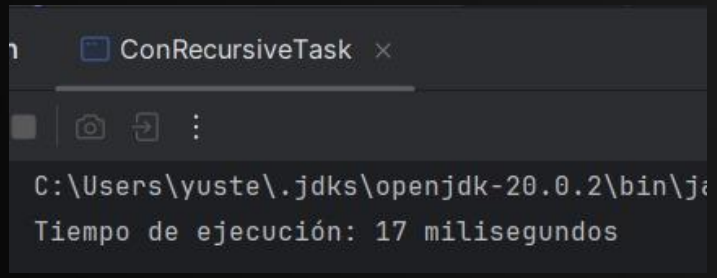
- Mejor rendimiento: Está diseñado para ser más eficiente en tareas paralelas.
- Facilidad de uso: Es más fácil de usar que Threads. ForkJoin utiliza un conjunto de clases y métodos que simplifican el desarrollo de tareas paralelas.
- Menos errores: Es menos propenso a errores que Threads. ForkJoin utiliza un modelo de programación más simple que el que utilizan los Threads.
- Tareas grandes: Es ideal para tareas que se pueden dividir en subtareas. Por ejemplo: Dividir una lista en listas más pequeñas.

DESVENTAJAS:

- Compatibilidad: No es compatible con todas las plataformas ya que es una API de Java 8 y no todas las plataformas soportan Java 8 (Por ejemplo: Windows XP)
- Extensión: Es más difícil de entender que Threads. Esto se debe a que es un framework más complejo que Threads
- Difícil de depurar: Dado que se pueden llegar a crear varias subtareas es difícil poder seguir adecuadamente el control sobre ellas.
- Menos eficiente que Threads: Con tareas pequeñas no llega a ser más funcional y eficiente que Threads.

EJEMPLOS DE USO

- Ejemplo con RECURSIVTASK



- Ejemplo sin RECURSIVTASK