Spring Batch

¿Qué es Spring Batch?

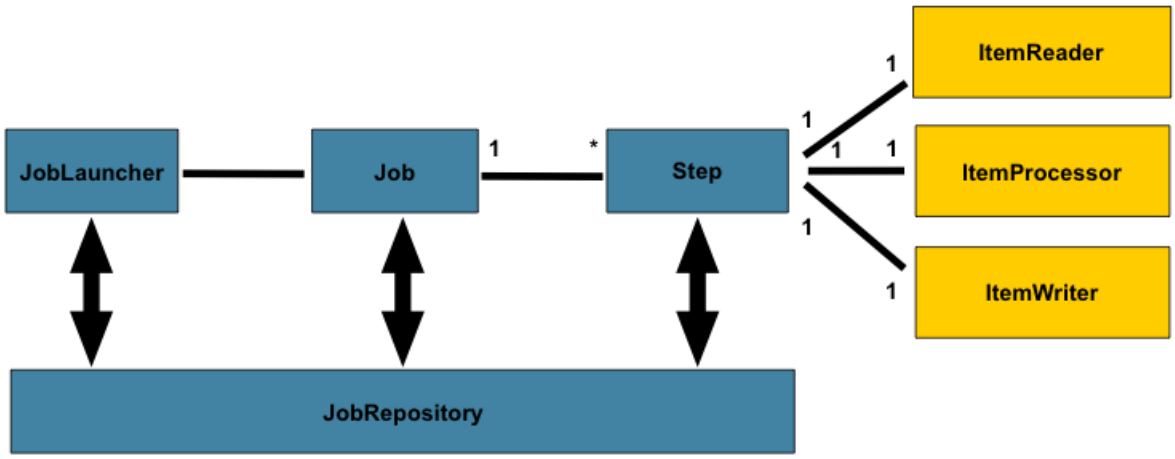
Spring Batch es un framework ligero y de código abierto desarrollado por la comunidad de Spring, destinado a facilitar la creación y gestión de **procesos batch**. Los **procesos batch** (o procesamiento por lotes) son un tipo de procesamiento de datos que se ejecuta en bloques o lotes, en lugar de tratar los datos de uno en uno.

Características principales

- **Ejecución automática**: Se inician en horarios predefinidos o intervalos regulares automáticamente, sin intervención manual.
- **Sin supervisión constante:** Una vez configurados, operan de manera autónoma y no requieren supervisión continua.
- **Carga y Volumen de Datos**: Pueden procesar grandes cantidades de datos de una sola vez, como actualizar miles de registros en una base de datos de un banco.
- **Impacto en el entorno transaccional**: Se ejecutan en horarios de baja actividad, como en la noche o fines de semana, para evitar interferir con las operaciones diarias.

Componentes de Spring Batch

Este framework está compuesto por los componentes representados en el siguiente diagrama:



- ❖ Job (Trabajo):
  - Es la tarea completa que se quiere realizar.
  - Puede tener uno o varios pasos (Steps).

- ❖ JobRepository (Repositorio de Trabajos):

- Guarda información sobre la ejecución de los *Jobs*.
- Ayuda a saber si un Job (tarea) se completó o falló.

❖ JobLauncher (Lanzador de Trabajos):
- Inicia la ejecución de un Job.

❖ Step (Paso):
- Es una fase específica dentro del Job.
- Un proceso (Job) debe tener, al menos, un step.
- Aunque **no es obligatorio**, un step puede estar compuesto de tres elementos:
  - ItemReader (Lector): Lee los datos de una fuente, como una base de datos o un archivo.
  - ItemProcessor (Procesador): Toma los datos del ItemReader y los transforma o procesa. No es obligatorio su uso.
  - ItemWriter (Escritor): Escribe los datos procesados en un destino, como otra base de datos o archivo. Si hay un ItemReader también debe haber un ItemWriter.

Chunk-Oriented Processing (Procesamiento Orientado a Bloques)

Chunk-Oriented Processing es un patrón de diseño utilizado en Spring Batch para procesar grandes volúmenes de datos de manera eficiente. Se basa en la idea de procesar los datos en "trozos" o chunks, en lugar de hacerlo registro por registro o todo a la vez.

¿Cómo funciona?

1. Inicio del proceso:
   - Los datos se dividen en bloques o "chunks" de un tamaño definido anteriormente.
   - Se inicia un Job en Spring Batch.
   - El Job contiene uno o más Steps que utilizan Chunk-Oriented Processing.

   Ejemplo: Un archivo con 10,000 registros se divide en chunks de 500 registros cada uno para su procesamiento.

2. Lectura del chunk (Read):
   - El ItemReader comienza a leer items de la fuente de datos.
   - Lee hasta alcanzar el número de items definido para el chunk.

   Ejemplo: El ItemReader lee 500 registros del archivo en cada operación.

3. Procesamiento del chunk (Process):
    - Cada item leído pasa al ItemProcessor (si está configurado).
    - El ItemProcessor transforma o enriquece cada item del chunk.

    Ejemplo: Se convierte el formato de fecha en los registros a un formato estándar.

4. Escritura (Write):
    - Una vez procesados todos los items del chunk, se pasan al ItemWriter.
    - El ItemWriter escribe todos estos items juntos en el destino.

    Ejemplo: Los 500 registros procesados se almacenan en una base de datos.

5. Transacción:
    - Todo el proceso de leer, procesar y escribir el chunk ocurre en una única transacción.
    - Si todo es exitoso, se hace commit de la transacción.
    - Si hay un error, se hace rollback de toda la transacción.

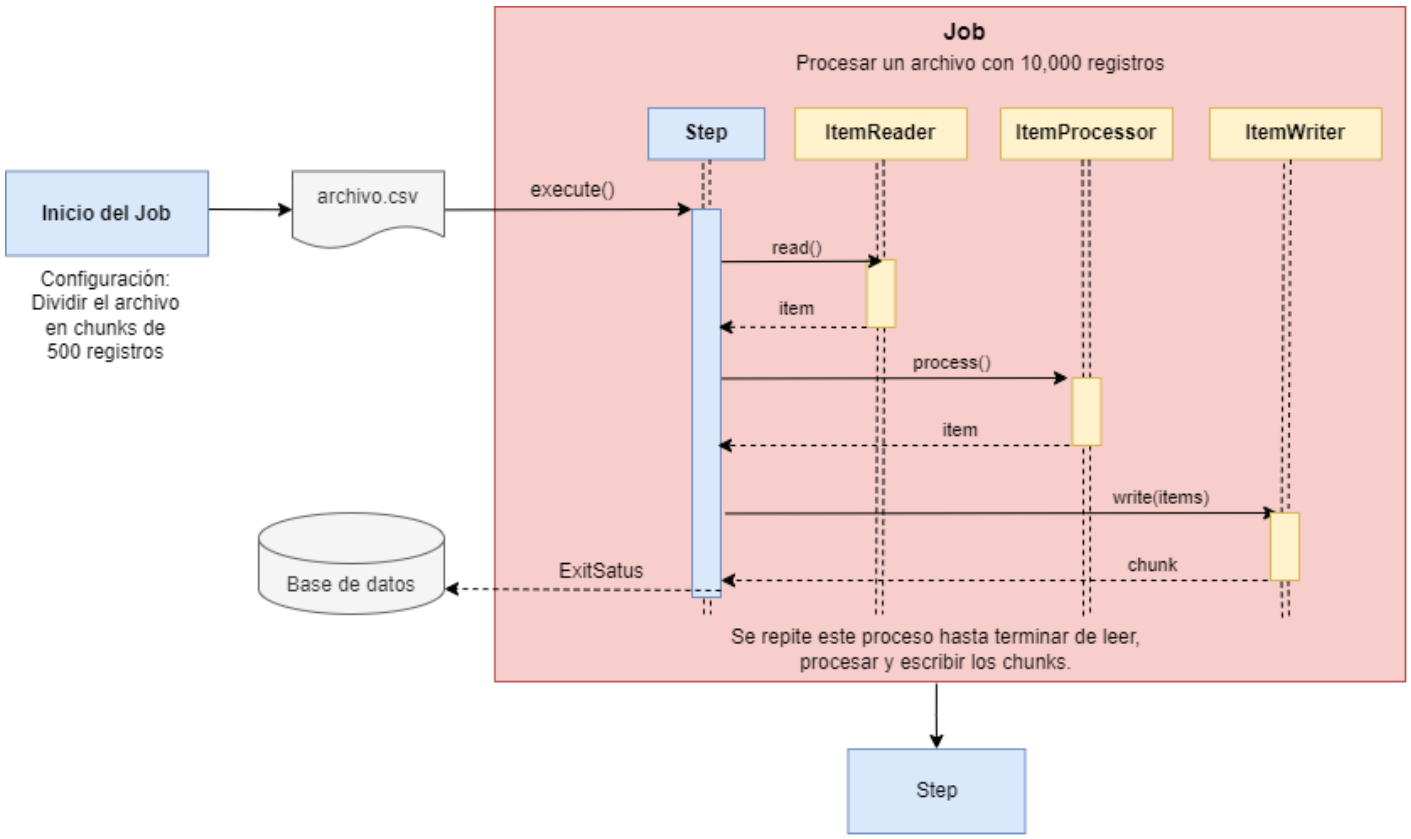
6. Repetición:
    - Después de completar un chunk, el proceso vuelve al paso 2 para el siguiente chunk.
    - Esto se repite hasta que se hayan procesado todos los datos.

    Ejemplo: Después de procesar un chunk de 500 registros, se pasa al siguiente chunk de 500 registros.

7. Finalización:
    - Cuando no quedan más datos para leer, el Step se completa.
    - Si hay más Steps en el Job, se pasa al siguiente.

8. Gestión de errores:
    - Si ocurre un error durante el procesamiento de un chunk, Spring Batch puede reintentar el chunk o saltar al siguiente, dependiendo de la configuración.

Al diagramar el anterior ejemplo quedaría algo así:



Ventajas:

- Procesa datos en lotes pequeños, evitando sobrecargar la memoria.
- Si falla un chunk, solo se pierde ese chunk, facilitando la recuperación.
- Permite optimizar el uso de recursos y cierto grado de paralelismo.

Desventajas:

- Complejidad adicional: Requiere más configuración y código que un procesamiento simple.
- Dificulta tareas que necesitan acceso a todos los datos simultáneamente.
- Múltiples transacciones pequeñas pueden impactar el rendimiento en algunos casos.