Spring Boot Spring Batch

Spring Batch

Framework de Spring para el procesamiento de datos.

Procesos batch

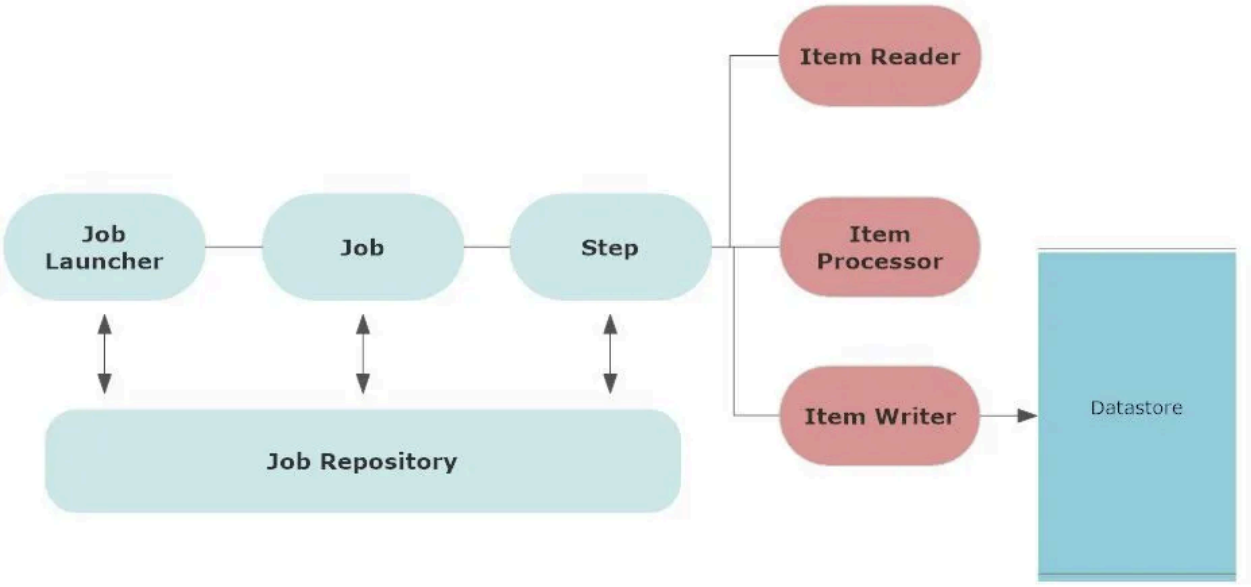
Es un proceso de grande lotes o volúmenes de información, involucra el consumo, procesamiento y transformación de datos para finalmente enviarlos a una nueva fuente de datos. Esto se realiza a través de un trabajo automático.

Suelen ser procesos planificados que se lanzan de manera automática sin necesidad de ningún tipo de intervención humana, al ser procesos pesados por la cantidad de información que manejan, es normal que se ejecuten en horarios de baja carga de trabajo para no interferir con otros procesos.

Spring Batch

es un framework de Spring para procesar grandes volúmenes de datos de forma eficiente y automatizada mediante tareas por lotes (batch processing).

Arquitectura de Spring Batch



Componentes:

- **JobRepository**

Guarda el estado y la historia de ejecución de los **Jobs** y **Steps**.

- **JobLauncher**

Es el encargado de lanzar o iniciar los procesos, es decir de la ejecución del **Job.** Éste puede ser programado o activado manualmente.

- **Job**

Es el proceso que se ejecutará con los lotes de información con los que se cuenta. Un **job** podría estar conformado de uno o muchos **steps.**

- **Step**

Un **step** es una tarea específica del proceso de **job**, por ejemplo, leer los datos, procesarlos o escribirlos en otra base de datos. Cada step usa un **ItemReader,** un **ItemProcessor** y un **ItemWriter.**

- **ItemReader**

Se encarga de leer los datos desde la fuente en la que se encuentren de manera inicial (base de datos, archivo CSV, API, archivo .txt, Excel, etc.).

- **ItemProcessor**

Procesa los elementos, filtra y transforma los datos, de acuerdo a lo que se le haya solicitado. Por ejemplo, podría filtrar ciertos datos, como seleccionar de una lista las filas con ciertas características, determinada edad, género, nacionalidad, o bien, realizar modificaciones como podría ser convertir mayúsculas a minúsculas, calcular valores, validar datos.

- **ItemWriter**

Se encarga de guardar los datos en un destino (otra base de datos, archivo, etc.).

Proceso:

1. El proceso iniciaría siendo activado de manera manual o a de manera automática, siendo el **JobLauncher** quien se encarga de ello.
2. **Job** establecerá el proceso completo de ejecución y definirá la secuencia de pasos (**steps**) que deben ejecutarse. Recordemos que cada step tiene un **ItemReader,** un **ItemProcessor** y un **ItemWriter.**
3. El **ItemReader** leer el archivo o base de datos donde se tenga toda la información con la que se trabajará.

4. El **ItemProcessor** procesará todos los datos y los filtrará o transformará de acuerdo a la instrucción recibida.
5. El **ItemWriter** guardará los datos procesados en la base de datos.
6. Al finalizar, se guardará de manera automática información sobre los jobs y steps ejecutados, por ejemplo:

    ✅ Estado del Job (COMPLETED, FAILED, RUNNING).
    ✅ Fecha y hora de inicio y fin de ejecución.
    ✅ Último Step ejecutado (para poder continuar si se interrumpe).
    ✅ Detalles de errores en caso de fallas.

Bibliografía:

- Maldiny. (s.f.). Spring Batch en Castellano [Repositorio GitHub]. GitHub. https://github.com/maldiny/Spring-Batch-en-Castellano
- Better Java Code. (s.f.). Step by Step Spring Batch Tutorial. Recuperado el [07-febrero-2025], de https://betterjavacode.com/spring-boot/step-by-step-spring-batch-tutorial

Astrid Itzel Alcántar Zárate