Natalia Denise Fuentes Rubio
Semana 2 – Evidencias
Academia Java

Tarea 2: Realizar una investigación sobre *Spring Batch*

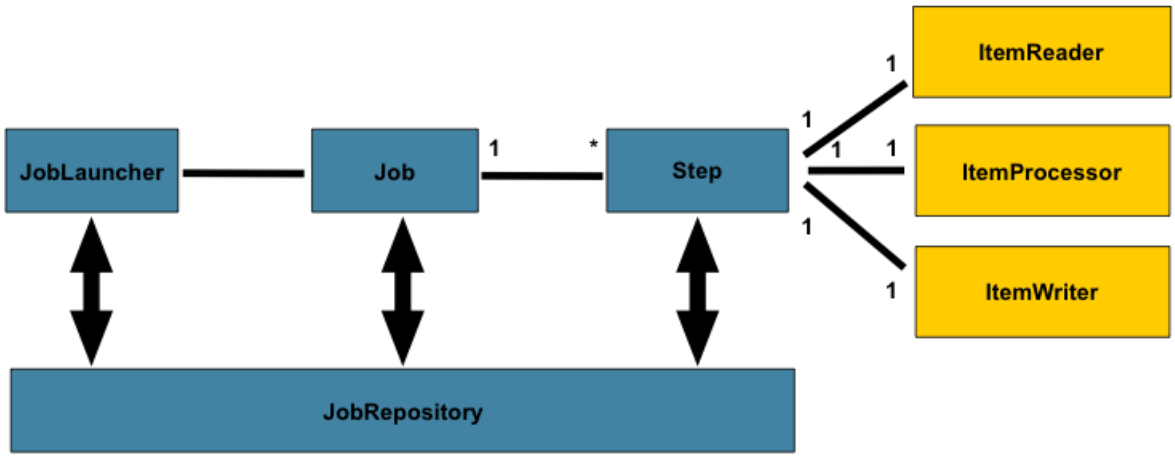
Los procesos Batch o procesos por lote, son programas que suelen ser ejecutados sin ningún tipo de intervención humana. Se manejan procesos relativamente pesados, ya que, vinculan un número considerable de información y su ejecución suele ocurrir cuando la carga o el procesamiento de información es bajo, con el fin de no interponerse entre las transacciones. Por ejemplo; generación de facturas o migración de bases de datos a otra.

Spring Batch es un framework ligero totalmente enfocado en los procesos Batch. Además de marcar unas directrices para el diseño de procesos, proporciona con una gran cantidad de componentes que intentan dar soporte a las diferentes necesidades que suelen surgir a la hora de crear estos programas: trazas, transaccionalidad, contingencia, estadísticas, paralelismo, particionamiento, lectura y escritura de datos, entre otros.

Dentro de sus funcionalidades tenemos:

- Administración de la transacción
- Procesamiento en Chunk
- Declaración de I/O
- Iniciar/Detener/Reiniciar los procesos
- Volver a intentar / Omitir algún paso o tarea
- Administración web (Solo disponible a través de Spring Cloud Data Flow)

Spring Batch está compuesto por varios componentes que se ejemplifican en la siguiente imagen y se da una breve explicación de cada uno de ellos.



JobRepository: Encargado de la persistencia de metadatos relativos a los procesos tales como qué procesos hay y cuántos activos.

JobLauncher: Encargado de lanzar los procesos suministrando los parámetros de entrada solicitados por cada uno de los procesos.

Job: Representación de un proceso, es el contenedor de uno o más pasos; debe tener por lo menos un step.

- JobInstance: Representación lógica de un determinado *job* con ciertos parámetros de ejecución.
- JobParameters: Conjunto de parámetros implementados para comenzar la ejecución, identificarla o proporcionar datos de la misma.
- JobExecution: Ejecución de una determinada instancia del *job* en un tiempo determinado.

Step: Elemento independiente que representa una de las fases que componen algún proceso.

Los steps pueden estar compuestos por tres elementos:

- ItemReader: Lee los datos de una fuente de datos
- ItemProcessor: Trata la información obtenida por el reader (paso opcional)
- ItemWriter: Guarda información que haya tratada por el Processor o que haya sido leída por el Reader. Si existe un Writer, existe un Reader.

Step encapsula cada una de las fases de un *Batch*. De este modo, el Batch está compuesto por uno o más *steps*, los cuales pueden ser tan simples o complejos como el desarrollador lo determine.

El StepExecution representa cada intento de ejecución de un cierto *step*.

A su vez, el *StepExecution* esta formado por un ExecutionContext que contendrá la información que se determine oportuna para la ejecución del *step* como estadística o información necesaria del estado del *Batch;* de aquí se destacan los siguientes campos:

- Status: Indica el estado en el que se encuentra el *step.* Puede ser *Started, Failed* o *Completed.*
- exitStatus: Contiene el código de salida del *step*. Puede ser recuperado en cualquier punto de la ejecución del *Batch.*

Otros elementos a tener en cuenta en la definición de flujos entre steps:

- Tag end: Determina la finalización inmediata del *job* tras cumplir su condición. *ExitStatus* y *BatchStatus* en estado *Completed*.
- Tag fail: Determina la finalización inmediata del *job* tras cumplir su condición. *ExitStatus* y *BatchStatus* en estado *Failed.*
- Tag stop: Determina la parada inmediata del *job* tras cumplir su condición. *BatchStatus* en estado *Stopped.*

Por su parte, el JobRepository es el mecanismo de persistencia para todos los elementos que forman un *Batch,* provee operaciones para la gestión del *JobLauncher, Job* y los *steps* del *Batch*.

El JobLauncher representa una interfaz para lanzar ejecuciones de un *job* con un conjunto de parámetros como entrada. Puede tener ejecución síncrona o asíncrona.

La parte más interesante del diseño reside en los steps. A continuación, se describirá el enfoque *Chunk – Oriented* en el que se basa Spring Batch para la ejecución de los steps.

El proceso *Chunk – Oriented* inicia con el *ItemReader* que lee una cantidad de datos y los convierte en un *chunk*. Si hay un *ItemProcessor* definido, el *chunk* pasa por dicho proceso para ser tratado, esto se realiza dentro de un límite transaccional – se leen y se tratan tantos *chunks* como se quiera antes de pasar por el *ItemWriter* -.

Un **Tasklet** es un objeto que contiene cualquier lógica que será ejecutada como parte de un trabajo, se construyen mediante su interfaz y son la forma más simple para ejecutar un código.

A través del control de flujo de los *steps* se puede definir la lógica de negocio en función del estado de salida de otros *steps.*

Una de las mayores ventajas de Spring Batch es que se pueden manejar grandes volúmenes de datos de una forma eficiente, además si llegase a ocurrir un error no se pierde el trabajo realizado, ya que después de cada *chunk* se realiza un commit, por lo que se puede revertir el procesamiento a partir de un chunk, lo cual minimiza el impacto de la falla y facilita la recuperación.

Referencias

Abujas, D. R. (2023, 4 diciembre). Introducción a Spring Batch: qué es, ventajas y ejemplo real. Profile Software Services. https://profile.es/blog/que-es-spring-batch-ejemplo/

Maldiny. (s. f.). GitHub - maldiny/Spring-Batch-en-Castellano: Ejemplos prácticos de Spring Batch. GitHub. https://github.com/maldiny/Spring-Batch-en-Castellano

Parola, Y. (s. f.). Hablemos de spring batch. https://somospnt.com/blog/311-que-es-spring-batch

Rodríguez, M. A. (2020, 7 enero). Introducción a Spring Batch - Adictos al trabajo. Adictos Al Trabajo. https://adictosaltrabajo.com/2013/07/17/introduccion-spring-batch/