Spring Batch

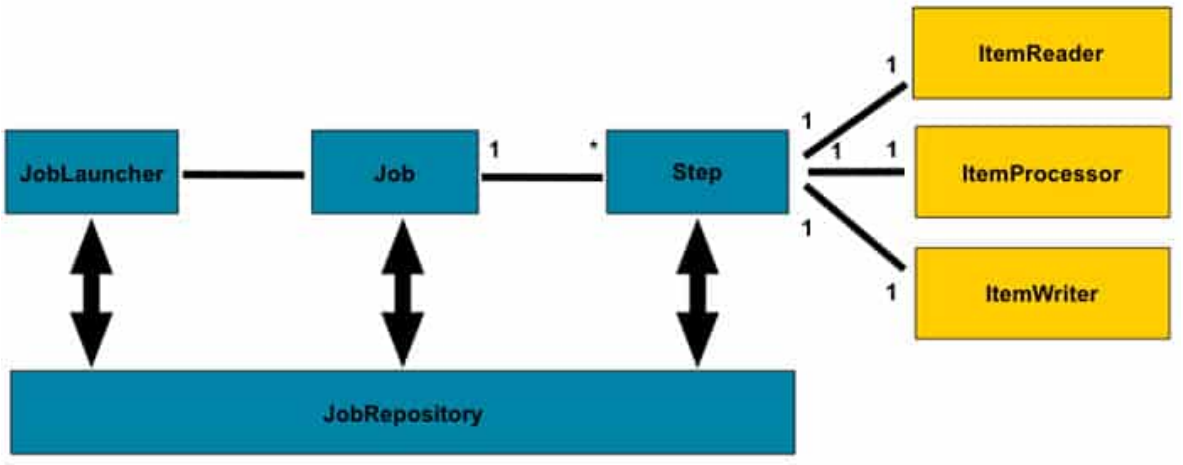
Spring Batch es un Framework módulo de Spring Boot, este Framework fue creado para procesamientos batch o por lotes, es decir, procesamiento de información por volumen.
Dicho Framework está enfocado al procesamiento batch, incluye herramientas que permiten monitorizar este tipo de procesos tan costosos, disponer de logs, configuraciones, transaccionalidad, estadísticas, alertas, etc.

¿Qué es un proceso Batch?

Un proceso Batch o proceso por lote, es un proceso pensado para trabajar con volúmenes muy grandes de datos y generalmente de una forma automatizada, es decir, sin intervensión humana.

Por ejemplo, si se tiene que cargar un archivo enorme, con millones de registros, o bien, un proceso nocturno, que a partir de una serie de consultas, envía una gran cantidad de e-mails, sms, etc, esto sería un proceso batch.

Este Framework se decribe por el siguiente diagrama:



Como se puede observar, Spring Batch está formado por diferentes elementos, los principales son.

Job Repository

Spring Batch está pensado para que la información de los procesamientos quede almacenada en un repositorio persistente, o bien, en memoria. Este repositorio se utiliza sobre todo para escritura, aunque también es consultado para comprobar si ya se ha procesado un fichero previamente. También se puede utilizar por si se produce un job fallido, para que en lugar de re-procesar todo el archivo de nuevo, únicamente se reprocese en trozo de información que ha fallado.

El JobRepository escribe y consulta una serie de tablas existentes en base de datos. Aquí tengo que hacer un inciso y es que la base de datos de este repositorio debe ser transaccional. Como se indica en esta pregunta de StackOverflow, MongoDB no se podría utilizar para el repositorio de Job.
Es responsabilidad de este repositorio almacenar información sobre cada job, step que se produzca, los parámetros del Job, los errores que tengan lugar, etc.

Job y Step

Un job es un bloque de trabajo y está compuesto por uno o varios pasos o steps. Una vez se han llevado a cabo todos estos pasos, se considera el job como completado.

Cada uno de estos steps suele constar de tres partes:

1. ItemReader: se encarga de la lectura del procesamiento por lotes. Esta lectura puede ser, por ejemplo, de una base datos; o también podría ser de un broker de mensajes o bien un fichero csv, xml, json, etc.

2. ItemProccessor: se encarga de transformar items previamente leídos. Esta transformación además de incluir cambios en el formato puede incluir filtrado de datos o lógica de negocio.

3. ItemWriter: este elemento es lo opuesto al itemReader. Se encarga de la escritura de los ítems. Esta puede ser inserciones en una base de datos, en un fichero csv, en un broker de mensajes, etc.

Si observamos, está centrado a trabajar con los ítems de manera unitaria. Para el procesamiento por lotes podemos definir de qué tamaño será el número de ítems en el que se organizará el procesamiento por lotes. Si cogemos un tamaño 20, leerá, procesará y escribirá de 20 en 20. Este número de ítems que se procesarán en cada uno de los commits que realice el step se denomina chunk.
Tasklet

Aunque no es obligatorio, un step puede estar compuesto de tres elementos: reader, writer y processor.

- **ItemReader:** Elemento responsable de leer datos de una fuente de datos (BBDD, fichero, cola de mensajes, etc…)
- **ItemProcessor:** Elemento responsable tratar la información obtenida por el reader. No es obligatorio su uso.
- **ItemWriter:** Elemento responsable guardar la información leída por el reader o tratada por el processor. Si hay un reader debe haber un writer.

Probablemente, la parte más interesante de todo este diseño resida en los steps (o pasos que conforman un proceso). En el siguiente apartado veremos cómo Spring Batch se basa en el enfoque «Chunk-Oriented» para la ejecución de los steps de un proceso.

Chunk-Oriented Processing

Chunk-Oriented es la técnica que utiliza Spring Batch para la ejecución de las fases de un proceso (nótese que es posible no utilizar en todos los steps este enfoque). Funciona de la siguiente manera.

El reader lee una porción de datos de la fuente de datos y los convierte en un «chunk» entidad que representa esa porción de información leida. Si existe un processor, ese chunk pasa al processor para que lo trate. Todo esto se realiza dentro de un límite transaccional o, lo que es lo mismo, leemos y tratamos tantos chunks como queramos antes de que sean persistidos por el writer.

Supongamos que tenemos un fichero en texto plano, donde debemos tratar cada línea para luego persistir cierta información en una base de datos. Si configuramos nuestro step con un intervalo de commit fuese igual a 10, lo que haría sería leer una línea del fichero, luego tratarla, leer otra línea del fichero, volver a tratarla, así hasta 10 veces. Una vez que ya hemos leído y tratado 10 líneas, el writer recibe esa información (los 10 chunks) y los persiste en base de datos. Este proceso se repetiría hasta terminar con todas las líneas el fichero.

La siguiente figura ilustra lo que hemos comentado en este punto: