Spring Batch

Spring es un framework de código abierto para el desarrollo de aplicaciones en Java. Proporciona características y funcionalidades para facilitar el desarrollo de aplicaciones empresariales complejas. Aborda situaciones como la gestión de dependencias, la inyección de dependencias, la integración de diferentes componentes y tecnologías, la configuración de la aplicación y la gestión de transacciones.

Spring proporciona modulos adicionales, como *Spring MVC* para desarrollo web, *Spring Data* para el acceso a datos, *Spring Security*, para la gestión de la seguridad de la aplicación, y *Spring Batch*, para el procesamiento de datos en lotes.

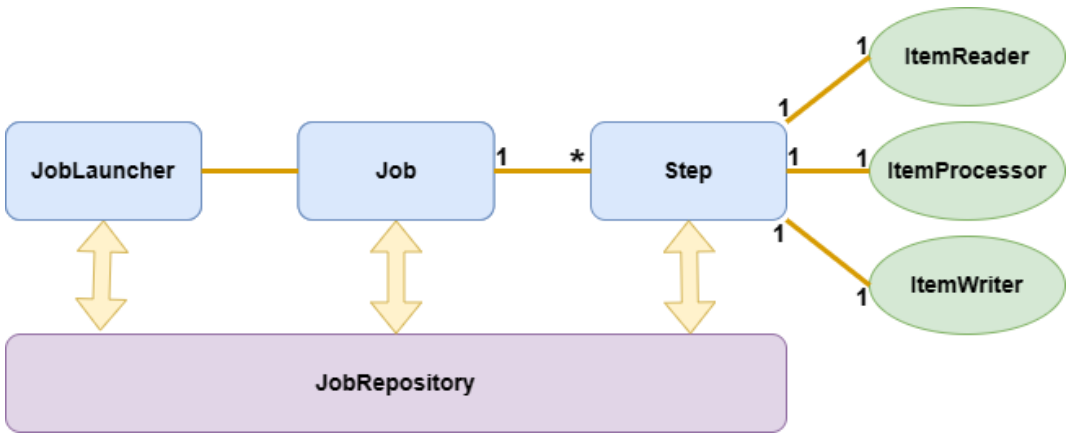
Spring Batch fue creado originalmente por el equipo de desarrollo de SpringSource, liderado por Dave Syer, como un proyecto en el marco de trabajo Spring en el año 2007. Fue lanzado como un proyecto de código abierto en el año 2008 bajo la Licencia Apache 2.0.

Como se mencionó anteriormente, Spring Batch es un módulo de Spring, desarrollado como una extensión de Spring para manejar el procesamiento de datos en lotes, proporcionando características como la lectura de datos desde una variedad de fuentes, el procesamiento de datos, la escritura de datos a diferentes destinos, la gestión de transacciones y el control de errores y recuperación. Además, ofrece funciones de administración y monitoreo de lotes, lo que facilita la gestión de las tareas de procesamiento en lotes.

Comúnmente Spring Batch es utilizado en aplicaciones empresariales que necesitan procesar grandes volúmenes de datos de manera programada y automatizada, como la generación de informes, la carga de datos en sistemas de gestión empresarial (ERP) o la limpieza y transformación de datos antes de su carga en un data warehouse.

Componentes principales de Spring Batch

Spring Batch está compuesto por una serie de componentes que proporcionan aspectos fundamentales en su funcionamiento, los cuales pueden verse en el diagrama a continuación:



JobLauncher: Es el componente encargado de lanzar los procesos suministrando los parámetros de entrada deseados. El JobLauncher es responsable de crear una instancia de Job y de iniciar su ejecución. Puede ser invocado de manera programática, a través de una API, o mediante el uso de herramientas externas como Spring Boot. Es capaz de ejecutar múltiples instancias del mismo Job de forma simultánea, lo que permite procesar grandes volúmenes de datos de manera eficiente.

Job: El Job es la representación del proceso. Un proceso, a su vez, es un contenedor de pasos (Steps). Una vez que se han llevado a cabo todos los pasos, se considera el Job como completado. Por ejemplo, se puede tener un Job que procese un archivo de datos y lo cargue en una base de datos.

Step: Un step (paso) es un elemento independiente dentro de un Job (un proceso) que representa una de las fases de las que está compuesto dicho proceso. Un proceso (Job) debe tener, al menos un step. Un step es una unidad de procesamiento más pequeña que pueda ser reutilizada en diferentes Jobs. Cada Step tiene un conjunto de tareas específicas que se ejecutan en una secuencia determinada. Por ejemplo, un Step puede leer datos de un archivo, procesarlos y escribirlos en una base de datos. Cada uno de estos Steps suele constar de 3 partes:

- ItemReader: Se encarga de la lectura del procesamiento por lotes. Lee los datos en pequeños lotes y los pasa al siguiente componente para su procesamiento. Por ejemplo, se puede utilizar un ItemReader para leer los datos de un archivo de texto, csv, xml, json, o de una base de datos.
- ItemProcessor: Se encarga de transformar ítems previamente leídos. Esta transformación además de incluir cambios en el formato puede incluir filtrado de datos o lógica de negocio. Por ejemplo, se puede utilizar un ItemProcessor para validar los datos de entrada o realizar operaciones de transformación.
- ItemWriter: Es un componente que se encarga de escribir los datos procesados en un destino específico, como una base de datos, un fichero csv, un bróker de mensajes, etc.
- Tasklet: Un step no tiene que estar compuesto por un reader, processor y writer. También puede tener únicamente una lógica de negocio. Es el caso del tasklet con el código que se desea ejecutar en el step.

Un Job 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.

JobRepository: Es un componente que se encarga de gestionar el estado del Job y de los Step durante su ejecución. El JobRepository almacena los metadatos sobre el estado actual del Job y de los Steps, como los datos leídos y los resultados procesados. También proporciona la capacidad de reiniciar un Job o un Step si se produce un error durante su

ejecución. El JobRepository escribe y consulta una serie de tablas existentes en base de datos transaccional.

Principios para definir un proceso batch

- Simplificar todo lo posible la lógica: de forma que quede fragmentada en procesos muy pequeños de lectura, procesamiento y escritura.
- Utilizar los mínimos recursos posibles: ya que se va a procesar un enorme volumen de datos.
- Revisar y optimizar sentencias SQL: es esencial que las consultas estén optimizadas ya que se verá tanto en el redimiendo de la base de datos como en los tiempos de ejecución del Job.
- Utilizar comprobaciones checksum: esto es muy útil a la hora de generar ficheros de gran tamaño, incluir en el pie del fichero un contador con el número de registros e información del procesamiento realizado que puede ayudar a comprobar la integridad del fichero.
- Utilizar pruebas de stress con datos lo más realistas posibles.