Introducción a Spring Batch.

Spring batch es una implementación o un acercamiento con un Framework como Spring Boot con el lenguaje de programación Java, enfocado específicamente en la creación de procesos Batch.

Proporciona una gran cantidad de componentes que intentan dar soporte a diferentes necesidades que suelen surgir a la hora de crear estos programas: **trazas, transaccionalidad, contingencia, estadísticas, paralelismo, particionamiento, lectura y escritura de datos.**

Batch Processing.

Es un método que se ejecuta repetitivamente para extraer altos volúmenes de datos sin o con poca interacción del usuario, también, se debe utilizar para datos no continuos, es decir, que para ejecutarlo no necesitamos datos en tiempo real. Evitar el desbordamiento de memoria en Java, ya que, se puede tener una carga de miles de líneas de código y ocasionaría errores incontrolables. Es aquí donde entra el batch processing para poder tomas esa información de un lado, transformarla y darle formato y depositarla en otro lado.

Conceptos básicos del Batch Processing.

En todas las industrias y para todos los trabajos, los conceptos básicos del procesamiento por lotes siguen siendo los mismos.

Los parámetros escenciales incluyen:

- Quién está enviando el trabajo.
- Que programa se ejecutará.
- La ubicación de las entradas y salidas.
- Cuándo se debe ejecutar el trabajo.

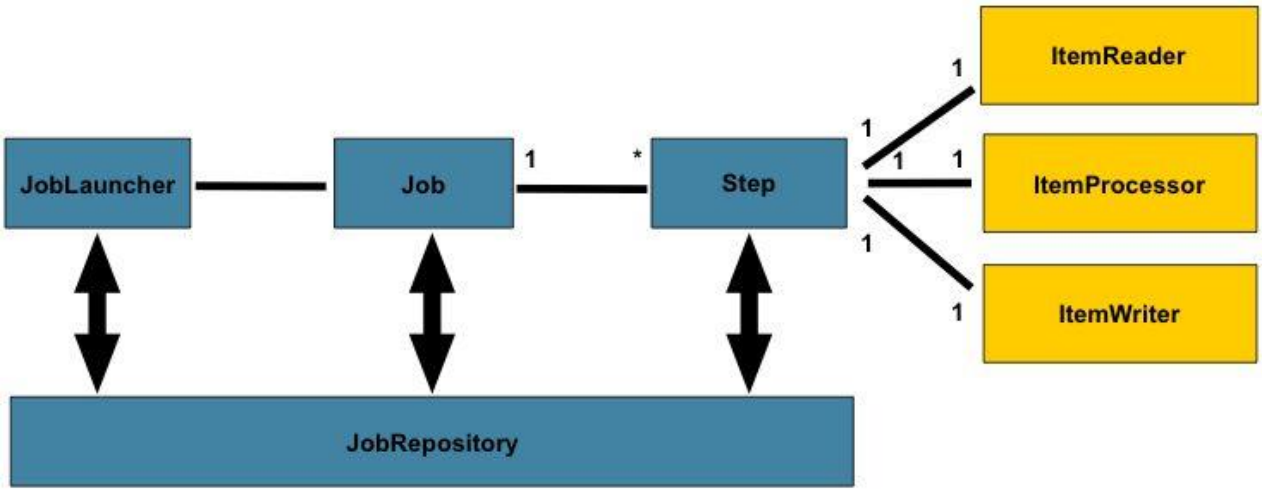
¿Quién?, ¿Qué?, ¿Dónde? Y ¿Por qué?

Ejemplo: Procesamiento de Datos Financieros

- Muchas empresas utilizan el procesamiento por lotes para automatizar los procesos de facturación.
- Una empresa mayorista solo puede facturar a sus clientes una vez al mes y pagar nómina a sus empleados cada dos semanas. Estos son ejemplos de procesamiento por lotes.

Ing. Ronaldo Tovar Reyes **03-05-2024**

Configuración de Chunk-Oriented Processing.



Al no tener datos continuos, no se deben introducir los miles de registros a Java, porqué hay pocos lenguajes que van a soportar el desbordamiento, entonces Spring Batch puede leer esos miles de registros por bloques, de cinco en cinco para validar los datos. Analizando posibles errores humanos de captura de información que se desea procesar.

JobLauncher: por medio de la inyección de dependencias, se encarga de ejecutar todos los **Jobs,** que se encuentren en el contexto de Spring Boot. En definición es el componente encargado de lanzar los procesos suministrando los parámetros de entrada deseados.

Job: es un proceso que se va a ejecutar en un tiempo determinado, dentro de este proceso se encuentran **Steps.** En definición, es la representación del proceso, que a su vez, un proceso es un contenedor de pasos (**steps).**

JobRepository: es el componente que nos va a ayudar a utilizar los metadatos y almacena la información de los **Job y Steps**, podemos especificar fallo y tolerancia por error humano, haciendo que, los bloques o lotes se corrijan. En definición, es el componente encargado de la persistencia de metadatos relativos a los procesos, tales como procesos en curso o estado de las ejecuciones.

Step (paso): es un elemento independiente dentro de un **Job,** que sepresenta una de las fases de las que está compuesto dicho proceso. Un proceso **Job** debe tener al menos un **Step.**

ItemReader: es el encargado de leer la información de entrada.

ItemProcessor: se encarga de procesar de acuerdo al bloque o lote definido y cuando termine de leer todos los registros

ItemWriter: se encarga de escribir los datos procesados a un almacenamiento distinto.

Tasklet: contiene el código que se desea ejecutar dentro de un **Step**, un **Step** no tiene que estar compuesto por un reader, procesor y writer. También puede tener únicamente una lógica de negocio **(Tasklet)**.

Conclusiones.

En resumen **Spring Batch** junto con **Chunk-Oriented Processing** ofrece una solución robusta y escalable para el procesamiento por lotes, permitiendo a los desarrolladores manejar grandes volúmenes de datos de manera efectiva, garantizando la integridad de los procesos y mejorando la eficiencia en el desarrollo de aplicaciones **Batch.**