Luis Alberto García Reyes

**Introduccion a Spring Batch**



# ¿Qué es Java?

Lenguaje de programación Orientado a objetos, se utiliza en diferentes aplicaciones gracias a que soporta la multiplataforma

# ¿Qué es spring?

Herramienta que hace que el desarrollo de aplicaciones web y microservicios con Spring Framework sea más rápido y fácil

* Configuración automática,
* Un enfoque obstinado de la configuración
* Capacidad de crear aplicaciones independientes

# ¿Dónde se clasifica Spring batch dentro de spring?

Dentro de las muchas cosas que podemos hacer con Spring aquí tenemos las siguientes:

* Microservicios
* Reactive
* **Batch**
* Web apps
* Even Driven
* Serverless
* Cloud

# Contenido del Video

Batch processing es el core de todo. Es un método que se ejecuta repetidamente para extraer grandes volúmenes de datos de un target o de un input a un output.

*Spring Batch es una extensión de Spring Framework que proporciona soporte para el procesamiento de grandes volúmenes de datos en lotes. Permite la ejecución eficiente de tareas repetitivas, como el procesamiento de grandes conjuntos de datos, la importación/exportación de datos, la generación de informes, entre otros*

## Cuando usarlo

Si existe la necesidad de obtener mucha información y procesarla día con día es necesario usar batch procesisng

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

Los parámetros (características principales) esenciales incluyen

* Quien está enviando el trabajo
* Que programa se ejecutara
* La ubicación de las entradas y salidas
* Cuando se debe ejecutar el trabajo

En otras palabras, quien, que, ¿donde y por qué?

## ¿Cómo funciona?

Spring Batch utiliza el concepto de ETL (Extract, Transform, Load) que es un proceso utilizado en la integración de datos para mover datos desde sus fuentes de origen, transformarlos a un formato adecuado y luego cargarlos en un destino, como un almacén de datos o una base de datos.

* **Extracción (Extract**): En esta etapa, los datos de los registros (logs) se extraen de sus fuentes. Estas fuentes pueden ser archivos de registro (logs) generados por sistemas, registros de bases de datos, flujos de eventos en tiempo real, o cualquier otro origen donde se almacenen los registros.
* **Transformación (Transform**): Una vez que los datos de los registros se han extraído, pueden necesitar ser transformados en un formato más estructurado y útil. Esto puede incluir la limpieza de los datos, la conversión de tipos de datos, la agregación de información, el enriquecimiento de datos con información adicional, la eliminación de duplicados, entre otras transformaciones.
* **Carga (Load**): Después de que los datos de los registros se han extraído y transformado, se cargan en el destino final. Esto podría ser un almacén de datos, una base de datos relacional, un sistema de análisis de registros (logs) o cualquier otro sistema que requiera estos datos para su análisis posterior.

## ¿Dónde usarlo?

Cuando necesitamos un batch procesing que extrae información de una entrada la va a procesar de manera repetitiva con datos no continuos que no necesitamos en tiempo real y la deposita en una salida que si no la necesito en ese momento la deposito en un lote donde pueda estar disponible (método de contingencia).

Otros ejemplos de su uso:

* **Procesamiento de Grandes Volúmenes de Datos**: Cuando se necesita procesar grandes cantidades de datos que no pueden ser procesadas de manera eficiente en tiempo real, el procesamiento en lotes es la solución ideal. Esto puede incluir el análisis de registros de servidor, el procesamiento de datos de sensores, la limpieza y transformación de datos, entre otros.
* **Generación de Informes**: Muchas aplicaciones empresariales necesitan generar informes periódicos basados en datos almacenados en la base de datos. El procesamiento en lotes permite ejecutar consultas complejas y generar informes detallados de manera eficiente y programada.
* **Integración de Datos**: En entornos empresariales complejos, es común tener datos dispersos en diferentes sistemas. El procesamiento en lotes puede ser utilizado para integrar estos datos, transformarlos en un formato común y cargarlos en un sistema de almacenamiento centralizado.
* **Carga Masiva de Datos**: Cuando se necesita cargar grandes volúmenes de datos en un sistema, como la migración de datos de un sistema a otro o la carga inicial de una base de datos, el procesamiento en lotes puede ser utilizado para realizar esta tarea de manera eficiente.
* **Procesamiento de Transacciones Financieras**: En el sector financiero, es común tener que procesar grandes volúmenes de transacciones financieras de manera programada, como el cálculo de intereses, la generación de extractos de cuenta, la detección de fraudes, etc.
* **Procesamiento de Archivos**: Muchas aplicaciones necesitan procesar archivos de entrada de manera periódica, como archivos de registro, archivos de texto, archivos CSV, etc. El procesamiento en lotes puede ser utilizado para leer estos archivos, procesarlos y generar archivos de salida según sea necesario.

## Spring Batch

Spring batch es un microframework dentro de Spring. Te ayuda a agilizar el procesamiento de Spring Processing

Batch Processing es técnica mientras que Spring Batch Framework que implementa Spring para el Batch Prcoessing. Este proporciona 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, estadística, paralelismo, particionamiento lectura y escritura de datos, etc.

Diagrama

Descripción generada automáticamente

Change Oriented Processing (CHUNQ)

Lo usa Spring Batch para leer grandes datos de información en pequeños bloques…

El usuario ya no tendría que hacer el procesamiento de los datos, el framework automatiza el procedimiento, -Job Repository nos ayuda a diseñar un JOB (Proceso) que se ejecuta a través de Steps además el Job repoitory almacea la información de los steps y de manera detallada.

# Componentes de Spring Batch

* **JobRepository:** Es el componente encargado de la persisteancia de metadatos relativos a los procesos tales como procesos en curso o estados de las ejecuciones
* **JobLauncher:** es el componente encargado de lanzar los procesos suministrando los parámetros de entrada deseados
* **Job:** Es la representación del proceso. Un proceso a su vez es un contenedor de pasos
* **Step:** un paso es un elemento independiente dentro de un job que representa una de las fases de las que esta compuesto dicho proceso. Un proceso debe de tener al menos un paso

## Tasklet

Un step no tiene que estar compuesto po 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 "Tasklet" es una interfaz que define una única tarea unitaria dentro de un trabajo (job). Esencialmente, un Tasklet es una implementación de una tarea específica que se ejecuta dentro de un paso (step) del proceso de Spring Batch

# Problemas de Spring batch

Aunque Spring Batch es una herramienta muy poderosa y flexible para el procesamiento en lotes, como cualquier otra tecnología, puede presentar desafíos y problemas en su implementación. Algunos problemas comunes que podrías enfrentar al trabajar con Spring Batch incluyen:

* **Rendimiento Ineficiente:** Si no se diseña adecuadamente, un trabajo de Spring Batch puede experimentar problemas de rendimiento, especialmente cuando se manejan grandes volúmenes de datos. Esto podría deberse a una mala optimización en la lectura, transformación o escritura de datos, o a una infraestructura subdimensionada.
* **Gestión de Transacciones:** Aunque Spring Batch maneja la gestión de transacciones de manera automática en muchos casos, es importante comprender cómo se manejan las transacciones en tu trabajo de batch para evitar problemas de consistencia y concurrencia.
* **Manejo de Errores:** El manejo adecuado de errores es crucial en un proceso de batch. Spring Batch ofrece varias estrategias de manejo de errores, pero es importante implementarlas correctamente para garantizar que los errores se manejen de manera adecuada y que el trabajo no falle inesperadamente.
* **Diseño Complejo de Flujos:** A medida que los trabajos de Spring Batch se vuelven más complejos, el diseño de los flujos de trabajo puede volverse complicado. Es importante mantener los flujos de trabajo lo más simples y comprensibles posible para facilitar el mantenimiento y la depuración.
* **Escalabilidad:** Aunque Spring Batch es escalable, puede ser necesario realizar ajustes y optimizaciones adicionales para manejar cargas de trabajo muy grandes o distribuidas.
* **Pruebas:** La prueba de trabajos de Spring Batch puede ser un desafío, especialmente cuando se trata de pruebas de integración que involucran la ejecución de trabajos completos. Es importante establecer estrategias efectivas de prueba para garantizar la calidad del software.
* **Monitoreo y Administración:** Es esencial implementar un buen sistema de monitoreo y administración para supervisar el estado y el rendimiento de tus trabajos de Spring Batch, así como para gestionar y solucionar problemas de manera efectiva.

# Ejemplo

Supongamos que tenemos el siguiente código:

Texto

Descripción generada automáticamente

En este ejemplo, **JobLauncher** es un componente de **Spring Batch** que se utiliza para lanzar trabajos, y Job es el trabajo del que quieres obtener el estado. La anotación @Autowired se utiliza para inyectar estas dependencias.

La función **getJobStatus()** lanza el trabajo utilizando **jobLauncher.run()** y luego devuelve el estado del trabajo utilizando **jobExecution.getStatus().**

Este código se puede integrar en una aplicación Spring Boot y exponer como un **punto final REST** para que puedas consultar el estado del trabajo desde cualquier cliente que pueda hacer solicitudes HTTP GET.