Spring Batch

Procesamiento por Lotes

El procesamiento por lotes es un método utilizado por las computadoras para completar trabajos repetitivos y de gran volumen de datos de forma periódica. Es común en tareas que requieren un esfuerzo de computación intensivo, como copias de seguridad, filtrado y clasificación, y se realiza a menudo en horas de menor actividad.

Ventajas

Eficiencia: Requiere mínima interacción humana y permite procesar grandes cantidades de datos juntos, utilizando mejor los recursos de computación.

Automatización: Reduce errores humanos y permite la supervisión automática de problemas.

Escalabilidad: Permite procesar millones de registros juntos cuando hay mayor disponibilidad de recursos.

Funcionamiento

Configuración: Especificar detalles como el nombre del trabajo, procesos, ubicación de datos y tiempos de ejecución.

Tamaño del lote: Determinar el número de unidades de trabajo a procesar en una operación.

Gestión de recursos: Asignar recursos necesarios durante el periodo de gestión de lotes.

Procesamiento por lotes en el sector financiero

En los servicios financieros, el procesamiento por lotes es crucial para manejar eficientemente grandes volúmenes de transacciones y datos. Al final de cada día laboral, las instituciones financieras consolidan todas las transacciones realizadas durante el día en un solo lote. Este proceso de consolidación permite **verificar y reconciliar los registros internos con las transacciones externas, ajustar los saldos de las cuentas y generar informes diarios que resumen las actividades financieras.** Este método es más eficiente que el procesamiento en tiempo real, ya que puede llevarse a cabo durante la noche, reduciendo la carga en el sistema durante las horas pico y garantizando la consistencia y precisión de los datos.

¿Qué es Spring Batch?

Spring Batch es un framework de procesamiento por lotes de código abierto, poderosísimo y altamente configurable, diseñado específicamente para crear aplicaciones robustas y escalables que procesan grandes volúmenes de datos de manera eficiente.

¿En qué situaciones es especialmente útil Spring Batch?

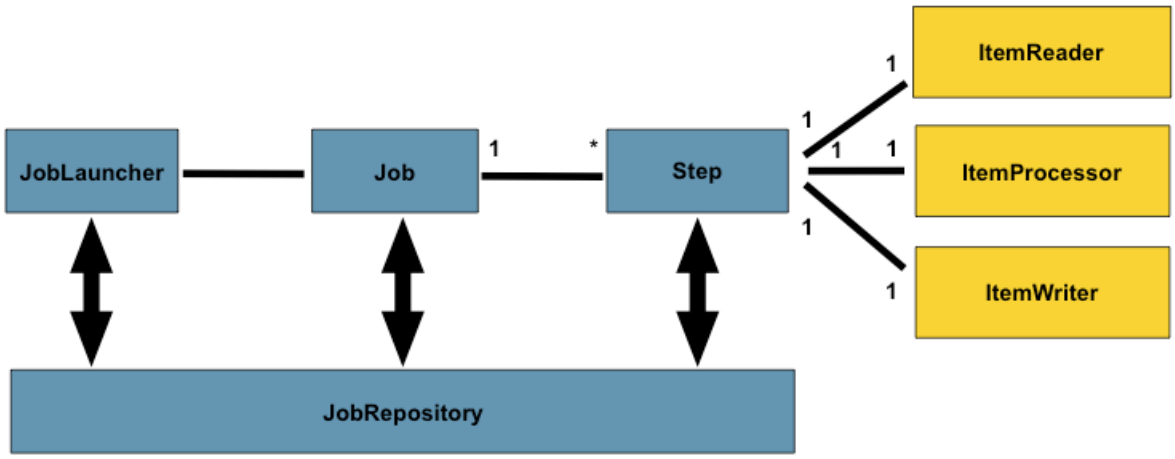
Procesamiento ETL (Extract, Transform, Load): Extrae datos de diversas fuentes, los transforma y los carga en un destino.

Integración de sistemas: Conecta diferentes sistemas y sincroniza datos entre ellos.

Automatización de tareas: Ejecuta tareas repetitivas de manera programada, como la generación de reportes diarios o la limpieza de datos.

Procesamiento de archivos: Lee y escribe archivos de gran tamaño, como archivos CSV, XML o JSON.

Componentes Principales



JobLaucher

Es el encargado de iniciar la línea de producción. Cuando recibe una solicitud, activa el Job. Actúa como un "disparador" que pone en marcha el proceso de batch.

Tiene una relación de uno a muchos con los Jobs, lo que significa que un JobLauncher puede iniciar múltiples Jobs.

Job

Un Job representa la unidad de trabajo completa. Es como una receta de cocina que define todos los pasos necesarios para preparar un plato. En Spring Batch, un Job define la secuencia de Steps que se deben ejecutar y los parámetros de configuración para el Job en su conjunto.

Contiene una colección de Steps, un JobRepository asociado y otros atributos como el nombre del Job, una descripción y una fecha de creación.

Un Job puede tener diferentes estados: STARTING, STARTED, COMPLETED, FAILED, UNKNOWN. El estado del Job se almacena en el JobRepository.

Step

Un Step es una unidad de trabajo más granular dentro de un Job. Representa una etapa específica del proceso, como leer datos de una base de datos, procesarlos y escribirlos en un archivo.

Contiene un ItemReader, un ItemProcessor y un ItemWriter. Opcionalmente, puede tener un Tasklet para realizar tareas personalizadas que no encajan en el modelo ItemReader-ItemProcessor-ItemWriter.

Spring Batch utiliza un enfoque basado en chunks para procesar los datos en lotes, lo que mejora el rendimiento y la gestión de transacciones. Un chunk es un conjunto de registros que se procesan de forma atómica.

ItemReader

El ItemReader es responsable de leer los datos de una fuente externa y proporcionarlos al ItemProcessor, un registro a la vez.

Existen diversos tipos de ItemReader, como:

- JdbcPagingItemReader: Lee datos de una base de datos utilizando paginación.
- FlatFileItemReader: Lee datos de un archivo plano (CSV, fixed-length, etc.).
- JpaPagingItemReader: Lee datos de una base de datos utilizando JPA.

Lee un chunk de registros a la vez y los devuelve al ItemProcessor.

ItemProcessor

El ItemProcessor es el corazón de la lógica de negocio de un Step. Recibe un item del ItemReader, lo procesa y devuelve un item transformado o enriquecido.

Realiza tareas como validación, cálculos, llamadas a servicios externos, etc.

ItemWriter

El ItemWriter es responsable de escribir los resultados del procesamiento en un destino externo.

Existen diversos tipos de ItemWriter, como:

- JdbcBatchItemWriter: Escribe datos en una base de datos utilizando batch updates.
- FlatFileItemWriter: Escribe datos en un archivo plano.
- JpaItemWriter: Escribe datos en una base de datos utilizando JPA.

Escribe un chunk de registros a la vez.

JobRepository

El JobRepository es el almacén persistente donde se guarda toda la información relacionada con los Jobs, los Steps y su estado de ejecución.

Permite reanudar Jobs interrumpidos, realizar auditorías, monitorear el progreso de los Jobs y proporcionar información histórica sobre las ejecuciones.

Utiliza una base de datos relacional para almacenar los datos.

Flujo de Ejecución

1. **Inicio**: Se invoca al JobLauncher para iniciar un Job específico.
2. **Ejecución del Job:** El JobLauncher delega la ejecución al Job.
3. **Procesamiento de Steps:** El Job ejecuta secuencialmente cada uno de los Steps que lo componen.
4. **Lectura de datos:** En cada Step, el ItemReader lee un item de la fuente de datos.
5. **Procesamiento de datos:** El ItemProcessor procesa el item y produce un resultado.
6. **Escritura de datos:** El ItemWriter escribe el resultado del procesamiento en el destino.
7. **Repetición:** Los pasos 4, 5 y 6 se repiten hasta que no queden más items para procesar o hasta que se cumpla alguna condición de parada.
8. **Finalización**: Una vez que todos los Steps se han ejecutado correctamente, el Job se marca como completado.