Introducción a Sprint Batch

Java es un lenguaje de programación multiplataforma y orientado a objetos.

Spring Boot

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

1. Configuración automática.
2. Enfoque obstinado de la configuración.
3. Tiene la capacidad de crear aplicaciones independientes.

En este se puede hacer servicio(microservicios) es la nueva normalidad que facilita el mantenimientos y escritura de código para tener aplicaciones más pequeñas y automatizadas.

Aplicaciones reactivas que son asíncronas que pueden soportar más llamadas de las que puede manejar y se realiza a través de hilos, que se utiliza con webflux.

- Cloud se utiliza para implementar sistemas distribuidos
- Web Apps facilita para realizar aplicaciones web app versión y sistemas bidireccionados basados en eventos
- Aplicaciones Event Driven son aplicaciones basadas en eventos funcionamiento al cambio.
- Serveles(funtional serves) que el sistema proveedor se ocupe d elo demás y el programador se enfoca en la lógica de negocio.

Sprint Bach (Batch Processing).

Es un método para ejecutar “trabajos” repetitivos de alto volumen de datos con poca o ninguna interacción del usuario. Es un método para un código que se quieres ejecutar muchas veces o mostrar mucha información, este sirve para no tener un desbordamiento de memoria.

Es agarrar información de un lado transformarla y darle un formato y pasarla a otro lado.

Parámetros esenciales:

- Quien esta enviado el trabajo.
- Que programa se ejecutara.
- La ubicación de las entradas y salidas (Tener un origen y asignar un lugar de origen).
- Cuando se debe ejecutar el trabajo (Tener una tarea para ejecutar datos)

Se utiliza el Batch Processing para realizar procesos donde utiliza grandes ejecuciones de procesos de datos no continuos. Ejecución de una tarea que tiene una entrada y una salida de manera repetitiva con datos no continuos (datos que no son en tiempo real).

Métodos de contingencia en Spring Batch: BackUp

Muñoz Alonso Gustavo

ETL(Extract, Transform And Load): Es un tipo de integración de datos que hace referencia a tres paso que se utilizan para mezclar datos de múltiples fuentes.

Que se utilizan a menudo para construir un almacén de datos, durante este proceso los datos se *extraen* de un sistema de origen y se *transforman* en un formato que se puede almacenar y se *carga* en un data warehouse u otro sistema, es cual es un enfoque alterno pero está relacionado y diseñado para canalizar el proceso a la base para mejorar el desempeño.

El término "Spring Batch" se refiere a un framework ligero de la plataforma Spring de Java que se utiliza para desarrollar y ejecutar procesos de batch, que también proporciona componentes para dar soporte a las necesidades a la hora de crear programas como son tazas, transacciones, contingencia, estadística, paralelismo, particionamiento, lectura, escritura de datos, etc

Un proceso de batch es una tarea automatizada que se ejecuta sin interacción directa del usuario, generalmente para procesar grandes volúmenes de datos en lotes o por lotes.

Spring Batch es un conjunto de características y herramientas para facilitar el desarrollo y la gestión de procesos de batch de manera eficiente y robusta. Algunas de las características clave de Spring Batch incluyen:

1. Configuración declarativa: Permite configurar los procesos de batch mediante archivos de configuración XML o anotaciones Java, lo que facilita la configuración y la comprensión del flujo del proceso.
2. División del trabajo en pasos: Permite dividir un proceso de batch en pasos más pequeños y manejables, cada uno realiza una tarea específica, como leer datos, procesarlos y escribir resultados.
3. Reintento de transacciones: Proporciona mecanismos para manejar transacciones y reintentar operaciones en caso de errores, que garantiza la integridad de los datos durante el procesamiento de lotes.
4. Escalabilidad y paralelización: que permite escalar los procesos de batch para manejar grandes volúmenes de datos, distribuyendo el trabajo entre varios procesadores o nodos en un entorno distribuido.
5. Gestión de errores y reintentos: tiene capacidades integradas para gestionar errores durante el procesamiento de lotes y reintentar automáticamente las operaciones fallidas, lo que mejora la robustez y la confiabilidad del proceso.
6. Integración con otros componentes de Spring: es integrar de manera transparente con otros componentes de Spring, como Spring Core, Spring Data, Spring Boot, entre otros, lo que facilita la creación de aplicaciones empresariales completas y coherentes.

Componentes principales

Para empresas que cuentan muchos registros lo que hace es distribuir todos los procesos en grupos más pequeños de registros el cual se le conoce como CHUNK.

Job Repository: (Job ->Step) Tiene un registro que almacena información de cómo funciona tus Job y Steps para verificar si se hicieron commit, skips o alguna otra acción y tiene tablas de manera detallada que fue lo que sucedió en el step o en un job. Conjunto de metadatos de procesos en curso o estados en ejecución.

Job: Procesos que se ejecutara en un tiempo determinado

Step: Pasos que están dentro de un Job. Este puede estar compuesto por un reader, processor y writer o simplemente tener la lógica de negocio (tasklet) el cual se utiliza para ejecutar un proceso agendado una sola vez.

Item Reader: Realiza la lectura los registros. Spring Batch cuando detecta un error te dice si quieres hacer un skip, tratar entre otras acciones. Lectura de procesamiento de lotes.

Item Processor: Procesa los registros uno por uno, y cuando procesa la acción encuentra algún fallo. Con el Spring Batch actuar. Transforma los ítems donde incluye cambios en el formato que puede incluir filtrado de datos o lógica de negocio.

Item Writer: Deposita los Step en una salida. Se encarga de la escritura de los ítems.

Job Launcher: Ejecuta Jobs dentro de contexto de Spring.

Servicio REST

Los servicios web REST *(Representational State Transfer)* son un estilo arquitectónico para diseñar sistemas distribuidos basado en estándares web, como *HTTP, URI y XML o JSON*. REST se centra en la simplicidad, la escalabilidad y el rendimiento, y es ampliamente utilizado en el desarrollo de aplicaciones web y móviles debido a su facilidad de uso y flexibilidad. Se refiere que REST es una interfaz para conectar varios sistemas basados en protocolos *HTTP* y sirve para generar operaciones y datos devolviéndonos datos en XML y JSON, haciendo mucho más fácil desarrollar una *API*. Los servicios Rest se basa en:

1. Recursos: En una arquitectura REST, los recursos son los objetos o datos que se pueden acceder a través de una URI (Uniform Resource Identifier). Cada recurso tiene su propia URI única, que se utiliza para identificar y acceder al recurso.
2. Verbos HTTP: Los servicios web REST utilizan los métodos estándar de HTTP (GET, POST, PUT, DELETE, etc.) para realizar operaciones CRUD (Create, Read, Update, Delete) en los recursos.
    a. GET se utiliza para recuperar datos
    b. POST para crear nuevos recursos
    c. PUT para actualizar recursos existentes
    d. DELETE para eliminar recursos.

    Se aplican para recursos a través de sus URI.

3. Representaciones: Los recursos en un servicio web REST pueden tener diferentes representaciones, como XML o JSON, que se utilizan para transmitir los datos entre el cliente y el servidor. Los clientes pueden especificar qué representación prefieren utilizando encabezados HTTP como "Content-Type" y "Accept".
4. Estado de la aplicación (Operaciones CRUD) : En una arquitectura RESTfull, el estado de la aplicación se transfiere entre el cliente y el servidor a través de las solicitudes y respuestas HTTP, en lugar de mantener un estado de sesión en el servidor. Esto hace que los servicios web REST sean altamente escalables y fáciles de implementar en entornos distribuidos. El cual se utiliza para realizar operaciones como crear, leer datos de recursos existentes, actualizar recursos y eliminar recursos.
5. Sin estado: Una de las características clave de REST es su naturaleza sin estado, lo que significa que cada solicitud HTTP contiene toda la información necesaria para procesarla y no depende de ninguna solicitud anterior. Esto simplifica la gestión del servidor y permite una mejor escalabilidad.
6. Interfaz Uniforma: Este la estandarización de las interacciones entre clientes y servidores, que incluye el uso consistente de URIs para identificar recursos, el uso de métodos HTTP estándar y el intercambio de representaciones de recursos.

Los servicios web REST son ampliamente utilizados en el desarrollo de aplicaciones modernas debido a su simplicidad, flexibilidad y compatibilidad con los estándares web existentes. Permiten una comunicación eficiente y escalable entre aplicaciones distribuidas y se han convertido en el enfoque dominante para diseñar *APIs* en la web. *El REST* no toma en cuenta los cambios anteriores, pero si tiene memoria cache, no es un *framework* y ni un *protocolo* y tampoco está atado a *un lenguaje de programación*.

Para construir un *API (aplicaciones de software interactivas)*, REST es la más utilizada para la construcción de proyectos que trabaja bajo un esquema cliente- servidor.

Una API es una abstracción de funciones y procedimiento.

REST es una lógica restricciones y recomendaciones bajo la cual se construye una API, es un estilo de arquitectura.

RESTFulAPI: es una API implementada y construida utilizando la lógica de REST.

Estas funcionan bajo una estructura cliente-servidor utilizando HTTP como protocolo de comunicación, en este el cliente utiliza una aplicación para hacer los requerimientos y el servidor hará las acciones que ejecutará.

Cada procedimiento de nuestra API contiene un Verbo HTTP, una dirección URI única, y la información necesaria que requiere el servidor para satisfacer el requerimiento incluyendo la información de autenticación del cliente.

Verbo HTTP: determinan el objetivo del cliente, el cual tiene estas funciones; GET (Devuelve información al cliente), POST (cliente crea recursos dentro der servidor, PUT/PATCH (Utilizar recursos dentro del servidor) y DELETE (Eliminar el cliente).

Muñoz Alonso Gustavo

CARACTERISTICAS DE REST

1. *Arquitectura Cliente – Servidor:* Separación de responsabilidad de cliente-servidor, así el cliente no se preocupa por el almacenamiento de datos y el servidor no se preocupa por la interfaz y el estado de esta, y su beneficio es la portabilidad y la interfaz puede ser multiplataforma, estos son independientes.
2. *Que sea Stateless:* La comunicación entre el cliente servidor no existe contexto, es decir cada petición entre estos componentes son independientes y no dependerá de peticiones anteriores, cada petición nueva se envía todo lo necesario al servidor. Toda petición será desechada a ser completada.
3. *Cacheability:* Guardar información en cache, posibilidad de que algunas respuestas sean almacenada y por cuanto tiempo.
4. Sistema basado en capas: El servidor debe estar compuesto por distintas capas cada una con una responsabilidad única y bien definida. Puede ser como vista, datos en cache, entre otras. Las capas deben de ser independientes para evitar errores, además solo deben de comunicar con capaz adyacentes y la comunicación entre estas debe de ser por la arquitectura de capas.
5. *Interfaz Uniforme:* Estandarización de la interfaz de comunicación entre el cliente-servidor.
6. *Extender la funcionalidad del cliente mediante envió de código OnDemand:* El servidor puede enviar script de código, para extender la funcionalidad de cliente. (Es opcional).