Spring Batch Tutorial

Description

Introduction

Generally all the applications have data in some or the other form which they have collected over time. Processing of large volumes of data faster is sometimes tedious. We would need robust transactional control and failure recovery mechanism which is complex to implement. In order to overcome this situation, Spring framework introduced a module called “Spring Batch”. Spring batch is used for batch processing. The processing of data can be done in the form of batch jobs. Spring batch provides reusable functions which we can use for processing large volumes of data. Spring batch provides features such as logging/tracing, transaction management, job processing statics, job restart, skip, and resource management.

Prerequisites to learning Spring Batch

Since Spring Batch is the module of Spring, you must know the basic concepts of Spring as well as Java language. Understanding of the databases as well as flat files is necessary because in most cases, they would be the source of data for processing. A basic brush up on Spring batch concepts or terminologies would be an add-on in learning Spring batch more quickly.

What is Batch processing

Batch Processing is a technique that is used for processing huge amounts of data with minimal human interaction. There are so many use cases wherein applications are processing millions or even billions of transaction everyday via batch processing.

Real life use cases of Batch processing.

Let’s consider you have developed a new banking application and it is mostly used to onboard new customers. The front facing client would enter the client details manually. But consider a case where some other bank wants to buy and use this application to onboard their existing customers. The customer data has been getting stored in the database since more than 10 years and you want to onboard all the customers in the new modern application. May be their data is store in some flat file or old datastore such as db2. It would be impractical to manually enter each record from existing large data to the new modern database such postgres or mongodb etc. In cases like these, Batch processing comes to rescue.

What is Spring Batch

Spring batch is the module of Spring framework that supports batch processing. Spring batch is used to create robust batch processing system.

Use cases where Spring Batch can be used

Reporting

Reporting can either be a monthly, daily, quarterly reporting. If you have an application where you need to process data for a specific timeline and view what's going on. In the banking domain, the reporting can be the amount of funds transferred or the total accounts opened.

Communication

Email sending can be a use case. You can process the loan database and see who all have missed their EMIs for the month. By looking at the data you can then send over email reminders.

Data migration

In order to migrate legacy data sitting in the old datastore to the new database so that modern applications can use it. Spring batch would do the migration job efficiently with minimal intereaction.

ETL task

ETL stands for Extract, Transform and Load. Sometimes the legacy data might be in the old format. Extracting them and transforming it to the new application standards is necessary. After transformation, the loading is done to the new datastore.

Parallel processing

To make the batch processing faster, Spring batch processes the records parallelly.

Features of Spring Batch

1. Spring batch has the concept of readers and writers. It has ItemReaders and ItemWriters to read from and write to destinations such as Flat file, XML database etc. Click here to read more about ItemReaders and ItemWriters
2. Spring batch stores all the information about the job execution metadata in the job repository.
3. We can skip or retry the processing of which ever records we want.
4. Spring batch provides the ability to start, stop or restart jobs.
5. Transaction management is possible in Spring batch. In case of any error the operation is rollbacked.
6. Spring team designed the framework in such a way that we can mainly focus on business rules rather than configuring most of the things.

Spring Batch Terminologies

Job

Job is defined as the complete batch process that is being executed. Job consists of multiple steps. We can make the step execute based on some condition.

Step

Step is a part of job which does the actual processing. There are two types of Steps i.e. Chunk based step and Tasklet based Step.

Chunk-based Step

Chunk-based step consist of ItemReader ItemWriter and ItemProcessor. We use Chunk based Step when we have a source for reading data and destination for writing data. We define a chunk size say 400. Those 400 records will be read, processed and written at once before taking another chunk. ItemProcessor can be used optionally to process or transform data. In most of the real world use cases Chunk-based Step is used.

Tasklet-based Step.

It has an interface with an execute method which keeps on running again and again until we tell the process to stop. Tasklets are usually used for operations like setup logics, stored procedures etc which canâ€™t be achieved without the box components.

ItemReader

ItemReader is used to read the data from the source.

ItemWriter

ItemWriter is used to write the data to the destination.

ItemProcessor

The role of ItemProcessor component comes after ItemReader and before ItemWriter. If you want to perform any custom processing on the read data such such as validations, filtering etc then we will use ItemProcessor.

Job Launcher

JobLauncher is used to start the actual job. We can also pass any additional parameters if any while starting the job. Mostly in production environment people start the job using scheduler.

Job Repository

JobRepository is like a store component which is used to store metadata information about the job. We don’t need to write any additional code for this, the metadata persistence is taken care by the framework. Information regarding JobExecutions, StepExecutions, JobParameters, and JobInstances are stored in the Job repository.

JobInstance

JobInstance is created by the Job launcher. The JobInstance is a combination of the job name and the job parameters.

JobExecution

A new JobExecution is created when we execute a JobInstance. The same instance of the job is executed when the same parameters are passed again. Whereas a difference JobInstance is executed when we execute the same job with the different parameters. Each time that we execute a JobInstance whether it is the same or different, we get a new JobExecution.

StepExecution

A new StepExecution is created when a step inside a job is executed.

List of ItemReaders and ItemWriters

In order to be proficient in Spring batch, know the available Item reader and Item writer is very important.

ItemReader

ItemReader is an interface which is used to retrieve data from the data source one item at a time for processing within the batch job. ItemReader interface consists of a method named read(). The framework has several implementations for reading data from database, files, message queues etc. Below are some of the important implementations of ItemReader.

KafkaItemReader	KafkaItemReader is used to reads messages from an Apache Kafka topic. It can be configured to read messages from multiple partitions of the same topic. This reader stores message offsets in the execution context to support restart capabilities.
FlatFileItemReader	FlatFileItemReader is used for reading and parsing flat files.
ListItemReader	Provides the items from a list, one at a time.

MongoItemReader	It is a Restartable ItemReader that reads documents from MongoDB via a paging technique
JsonItemReader	Reads items from a Json document

When using any of these ItemReaders, we will have to provide some specific configuration that will instruct the reader how to consume the items from the data store. There are more ItemReaders than the ones listed, click here to find the specific ItemReader according your use case.

ItemWriter

ItemWriter is used to write raw data or processed data to the destination. ItemWriter is an interface with has a single method named write(). Multiple data items are written in chunks (Example 500 records) at once instead of writing single item one by one. There are various types of ItemWriters provided by the framework which is used to write data to datastores like databases, Flat files, kafka topic etc. Below are some of the important ItemWriter.

KafkaItemWriter	KafkaItemWriter is used to write items to the default topic through the KafkaTemplate#sendDefault(Object, Object)Â method by using a Converter to map the key from the item.
FlatFileItemWriter	FlatFileItemWriter is used to write to flat files. To know more click here
MongoItemWriter	Given a MongoOperations object, items are written through the MongoOperations.save(Object) method.

There are more ItemWriters than the ones listed, click here to find the specific ItemReader according your use case.

ItemProcessor

ItemProcessor is used to process the data before writing to the destination. Spring batch provides ItemProcessor interface, which has a single process() method. Sometimes just reading the items from the source and writing to the destination is not sufficient. Applying Custom logic to the data is required before writing to the destination. Custom processing logic can be either filtering, validation, transformation etc.

Including ItemProcessor is purely based on use case. If use case deals with just moving the data from one source to another then, ItemProcessor is not required.

Conclusion

I hope you liked the post on Spring batch. Feel free to provide your thoughts in the comments section below.

Category

1. Spring batch

2. Spring Boot

Date Created
March 13, 2023
Author
kk-ravi144gmail-com