Spring Batch

--------------

Spring Batch provides advance services and features for high volume and high performance batch jobs using optimization and partitioning techniques.

It is highly scalable and can be used for processing of high amount of data.

A batch program reads a large number of records from a database, file, or queue,

processes the data based on the business needs, and then writes back data in the desired form.

Spring Batch Business Use Case

---------------------------

At the end of a month when a company has to send salary to its employee’s respective accounts.

Processing of salary slips at month end is when spring batch can be used.

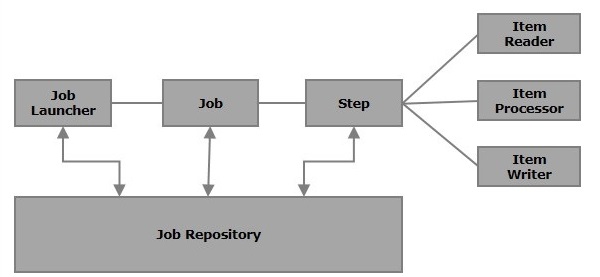
Sending out mass communication emails.

For generating automated reports on daily, weekly or monthly basis.

Executing business workflow automatically without human intervention.

Components of Spring Batch

The following illustration shows the different components of Spring Batch and how they are connected with each other.



### **Job**

In a Spring Batch application, a job is the batch process that is to be executed. It runs from start to finish without interruption. This job is further divided into steps

**Restartable** − In general, when a job is running and we try to start it again that is considered as **restart** and it will be started again. To avoid this, you need to set the **restartable** value to **false** as shown below.

<job id = "jobid" restartable = "false" >

</job>

### **Step**

A **step** is an independent part of a job which contains the necessary information to define and execute the job

### **Readers, Writers, and Processors**

An **item reader** reads data into a Spring Batch application from a particular source, whereas an **item writer** writes data from the Spring Batch application to a particular destination.

An **Item processor** is a class which contains the processing code which processes the data read into the spring batch. If the application reads **"n"**records, then the code in the processor will be executed on each record.

When no reader and writer are given, a **tasklet** acts as a processor for SpringBatch. It processes only a single task. For example, if we are writing a job with a simple step in it where we read data from H2 database and process it and write it to a file (flat), then our step uses −

* A **reader** which reads from H2 database.
* A **writer** which writes to a flat file.
* A **custom processor** which processes the data as per our wish.

<job id = "helloWorldJob">

<step id = "step1">

<tasklet>

<chunk reader = "h2Reader" writer = "fileWriter"

processor = "CustomitemProcessor" ></chunk>

</tasklet>

</step>

</ job>

### **JobRepository**

A Job repository in Spring Batch provides Create, Retrieve, Update, and Delete (CRUD) operations for the JobLauncher, Job, and Step implementations. We will define a job repository in an XML file as shown below.

<bean id=*"jobRepository"*class=*"org.springframework.batch.core.repository.support.JobRepositoryFactoryBean"*>

<property name=*"dataSource"* ref=*"dataSource"* />

<property name=*"transactionManager"* ref=*"transactionManager"*/>

<property name=*"databaseType"* value=*"h2"* />

</bean>

### **JobLauncher**

JobLauncher is an interface which launces the Spring Batch job with the **given set of parameters**. **SampleJoblauncher** is the class which implements the **JobLauncher** interface. Following is the configuration of the JobLauncher.

<bean id = "jobLauncher"

class = "org.springframework.batch.core.launch.support.SimpleJobLauncher">

<property name = "jobRepository" ref = "jobRepository" />

</bean>

## Mapper Class

The Mapper class, depending upon the reader, implements interfaces such as **row mapper**, **field set mapper**, etc. It contains the code to get the data from the reader and to set it to a Java class with **setter** and **getter** methods (Java Bean).

## Java Bean Class

A Java class with **setters** and **getters** (Java bean) represents data with multiple values. It acts as a helper class. We will pass the data from one component (reader, writer, processer) to other in the form of object of this class.

## Tasklet/processor

The Tasklet/processor class contains the processing code of the Spring Batch application. A processor is a class which accepts an object that contains the data read, processes it, and returns the processed data (in the form object).