Spring Batch
By
Apparao

Spring Batch Introduction:

Many applications within the enterprise domain require bulk processing to perform business operations in mission critical environments.

These business operations include automated, complex processing of large volumes of information that is most efficiently processed without user interaction.

These operations typically include time based events (e.g. month-end calculations, notices or correspondence), periodic application of complex business rules processed repetitively across very large data sets (e.g. Insurance benefit determination or rate adjustments), or the integration of information that is received from internal and external systems that typically requires formatting, validation and processing in a transactional manner into the system of record.

Batch processing is used to process billions of transactions every day for enterprises.

Spring Batch is a lightweight, comprehensive batch framework designed to enable the development of robust batch applications vital for the daily operations of enterprise systems.

Spring Batch is not a scheduling framework.

There are many good enterprise schedulers available in both the commercial and open source spaces such as Quartz, Tivoli, Control-M, etc.

It is intended to work in conjunction with a scheduler, not replace a scheduler.

Batch Job:

A Batch job reads data, processes it and writes processed data in specified format.

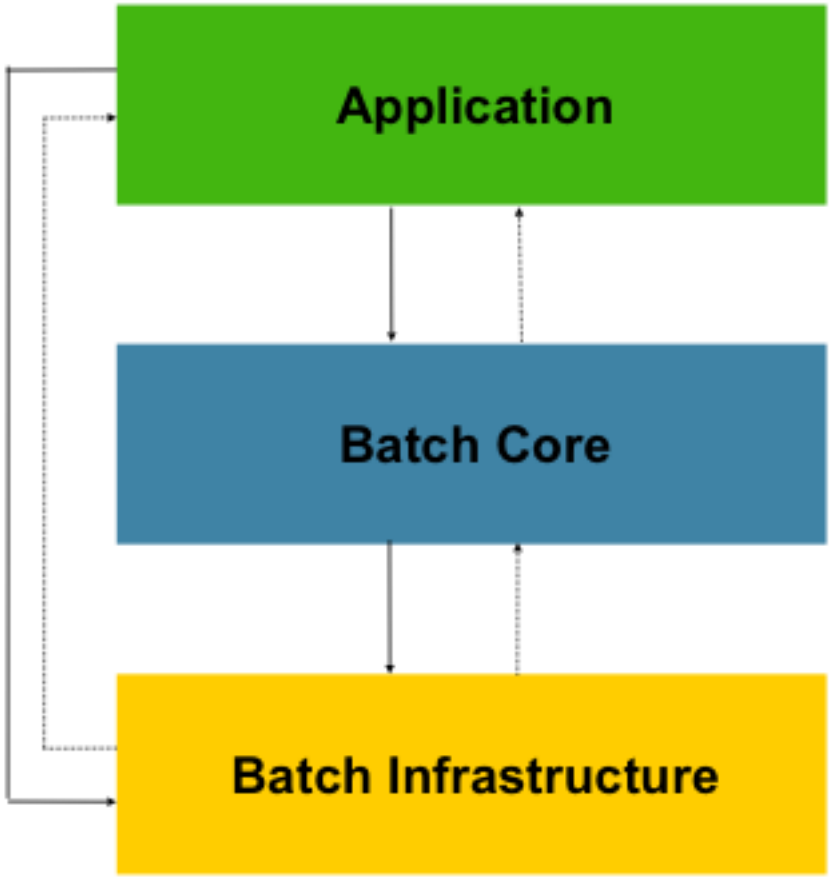
Examples:

- Health care:
  - Claim processing application – Batch job that runs daily to process incoming Claims, updates claim status as processed or rejected.
  - ID Car printing application – Batch application runs periodically (Say once in 3 Hours) to generate and print ID cards for newly enrolled Members.
- Banking / Finance:
  - Daily transaction report application – Daily Job that creates report out of transactions made for the day.
- Facebook:
  - Birthday Notification System – Daily job (Could be Cron job) that notifies users with their Friends' birthday.

Spring Batch Advantages

- **Infrastructure** – Spring Batch helps us to structure our code in a clean way by providing the necessary infrastructure that is used to implement, configure, and run batch jobs.
- **Chunk oriented processing** - Where items (data) are processed one by one and the transaction is committed when the chunk size is met. In other words, it provides us an easy way to manage the size of our transactions.
- **Error handling** - It provides awesome error handling.
  - For example, we can skip items if an exception is thrown and configure retry logic to decide whether our batch job should retry the failed operation.
- **Logging** – It gives perfect logging mechanism. We can trace out the steps execution in the persisted Database.

Spring Batch Architecture



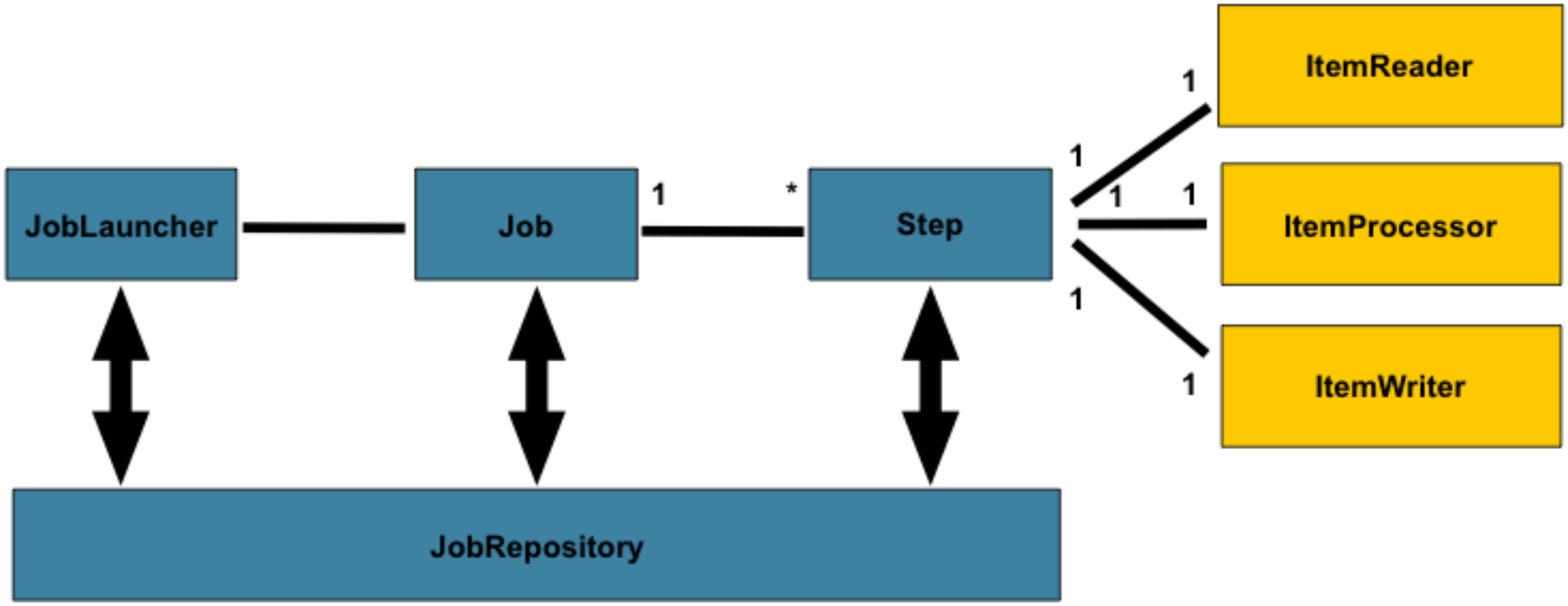
This layered architecture highlights three major high level components: Application, Core, and Infrastructure.

Application: The application contains all batch jobs and custom code written by developers using Spring Batch.

Batch Core: The Batch Core contains the core runtime classes necessary to launch and control a batch job. It includes things such as a JobLauncher, Job, and Step implementations.

Both Application and Core are built on top of a common infrastructure.

Batch Infrastructure: This infrastructure contains common readers and writers, and services such as the RetryTemplate, which are used both by application developers(ItemReader and ItemWriter) and the core framework itself.


JobLauncher
Job
1
*
Step
1
1
1
1
1
1
ItemReader
ItemProcessor
ItemWriter
JobRepository

JobLauncher:

JobLauncher represents a simple interface for launching a Job with a given set of JobParameters:

```
public interface JobLauncher {

 public JobExecution run(Job job, JobParameters jobParameters)
            throws JobExecutionAlreadyRunningException, JobRestartException;
}
```

It is expected that implementations will obtain a valid JobExecution from the JobRepository and execute the Job.

JobRepository:

It provides CRUD operations for JobLauncher, Job, and Step implementations.

When a Job is first launched, a JobExecution is obtained from the repository, and during the course of execution StepExecution and JobExecution implementations are persisted by passing them to the repository:

- Job – It denotes the actual Spring batch job. Each job is configured with one or more Steps.
- Step – It represents piece of code put together to handle the independent logical task.
  - For ex: Reading data from file – Is a step in Batch job
- ItemReader – It reads the input data and share each item one at a time to processor/writer for further processing
- ItemProcessor – It processes the incoming data from ItemReader and transforms it to the form that ItemWriter would understand
- ItemWriter – It writes data of an item to the output one at a time.

* ItemReader, ItemProcessor and ItemWriter – Configure individually to each Step.