**Spring Batch**

Spring Batch is a lightweight, Open Source batch processing framework designed to handle the day-to-day batch processing jobs involving bulk of data.

Batch processing is execution of series of job where each job may involve multiple steps.

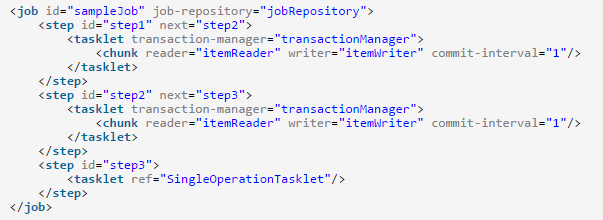
Each Step can work in two modes:

**Chunk Oriented Processing or READ-PROCESS-WRITE mode**: Step needs to read from a resource, process the data and then write back the process data to a resource. In this approach, Step has exactly one Item Reader(reader which reads from a resource, be it file, database, messaging queue etc.), Item Processor ( provides a hook to apply business logic) and Item Writer (writer which writes to a resource, be it file, database, messaging queue etc.).

**TASKLET mode**: Step has to perform a single operation (be it just sending email, executing a stored procedure, cleaning up files older than x days, etc.). In this approach Step includes a Tasklet interface which have only one method execute which can perform above mentioned activities.

A job is launched via Job Launcher, and Job Repository stores the metadata about the currently running process. Note that Steps can be chained together.

**Sample Configuration**



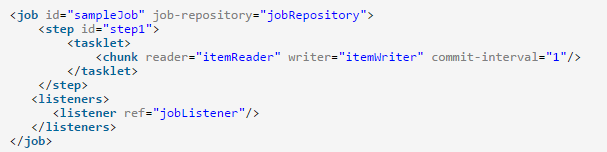
Above configuration reveal that there are three steps in our sample job. Step1 is working in Chunk Oriented mode that means it reads from a resource, process and then writes to resource. Step2 will be triggered once Step1 is completed (via next). Step2 as well is running in Chunk Oriented more, and will trigger Step3 once completed. Now Step3 is working in Tasklet mode that means it will perform a single operation (ref refers to a bean which implements Tasklet interface providing execute method implementation.

**Listeners**

Spring Batch also provides possibility to add Listeners on job and step level. Listeners provide the opportunity to perform a business logic before or after the job and step. For example if you would like to create a directory or move exiting data to another directory before performing the actual file processing job, you can use job listener which will provide the hook to be called just before the actual job starts and also a hook at the completion of job (to perform some other business logic if you want to do so).

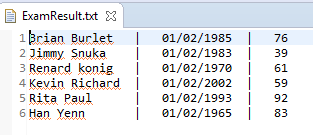
Listener on job level contains method beforeJob which will be called just before job starts, and afterJob which will be called just after job completed. Similarly, Listener on Step level contains methods beforeStep which will be called just before step starts, and afterStep which will be called just after step completed.

Below is a sample configuration with listener on Job level

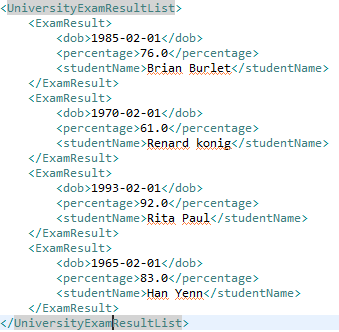


**Program 1**: Read a text file and write to an XML file (with listener)

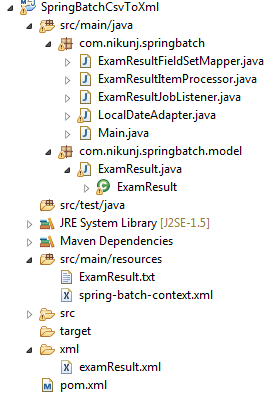
**Input:**

****

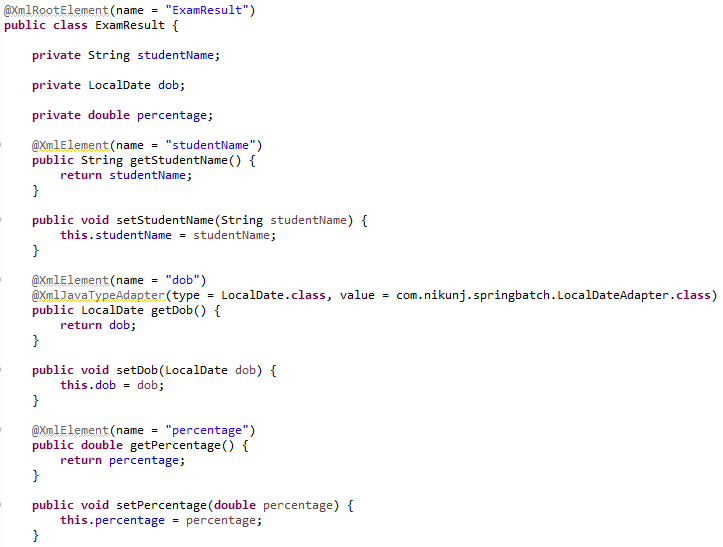
**Output:**

****

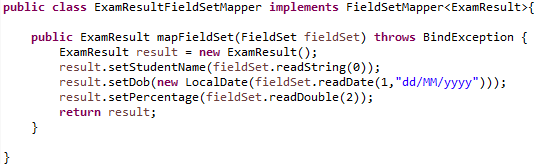
**Project Structure**

****

**Step 1:** Map the POJO with fields corresponding to the row content of above text file:

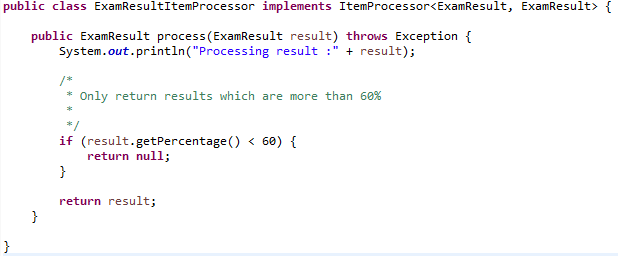
****

**Step 2:** Create a FieldSetMapper and it will be responsible for mapping each field form the input to a domain object

****

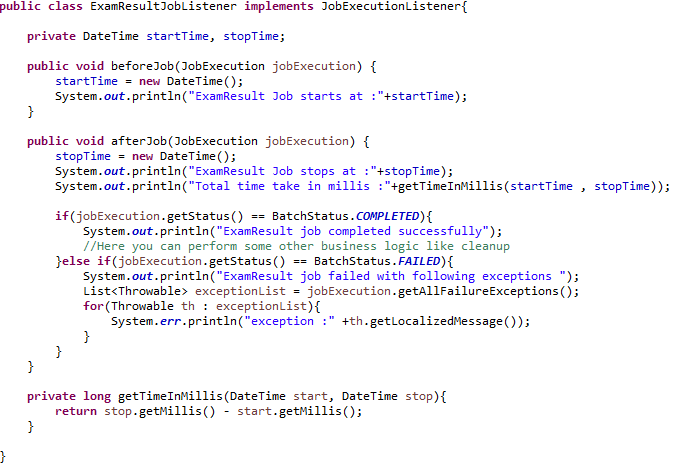
**Step 3:** Create an ItemProcessor

ItemProcessor is Optional, and called after item read but before item write. It gives us the opportunity to perform a business logic on each item. In our case, for example, we will filter out all the items whose percentage is less than 60.So final result will only have records with percentage >= 60.



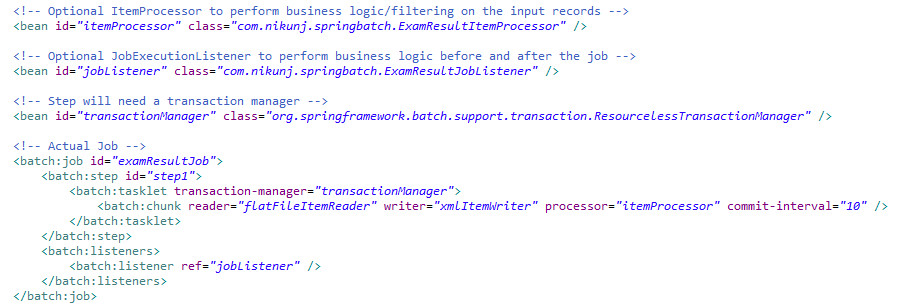
**Step 4**: Add a Job listener (JobExecutionListener)

Job listener is Optional and provide the opportunity to execute some business logic before job start and after job completed. For example setting up environment can be done before job and cleanup can be done after job completed.

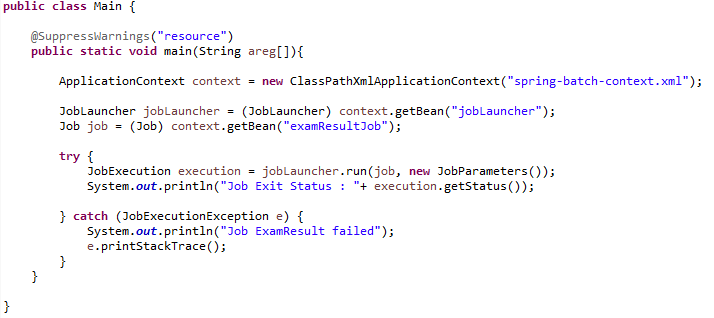


**Step 5**: Create Spring Context with job configuration



****

**Step 6:** Create Main application to finally run the job



**Program 2**: Read an XML file and write to MySQL Database

**Program 3**: Read From MySQL database & write to CSV file