|  |  |
| --- | --- |
|  | **Spring Batch** |
| Author: VLP/AHO | **S84 – Self-training roadmap** |
| Review: PHD |  |
| Date: 29.11.2016 | Imputation number: TODO |
| Version: 1.1 | Total scheduled time: 45h Total scheduled time for Coach: 2h Total scheduled time for Expert: 1h |

Self-Training for Spring Batch

**Table of contents**

1. Introduction 1

2. Prerequisites 1

3. References 2

4. Training plan 3

# 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 builds upon the productivity, POJO-based development approach, and general ease of use capabilities people have come to know from the Spring Framework, while making it easy for developers to access and leverage more advance enterprise services when necessary.

The purpose of this training is to give the trainee the basic knowledge of Spring Batch. After the training, trainee should be able to work on real projects which using Spring Batch as batch implementation.

# Prerequisites

Basic Java programming

Spring framework basic knowledge

JDBC knowledge

Maven basic knowledge

File processing in Java

Environment:

* JDK 1.8
* IntelliJ

# References

|  |  |
| --- | --- |
| **Ref** | **Source** |
| Spring Batch - Reference Document | <http://docs.spring.io/spring-batch/reference/htmlsingle> |
| Spring Batch in Action | <https://www.manning.com/books/spring-batch-in-action> |
| Spring Batch summary | <https://dzone.com/refcardz/spring-batch-refcard> |

# Training plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Type[[1]](#footnote-1)** | **Task** | **Description** | **Estimated Time (h)** |
|  | R | Basic understanding | Read **Spring Batch Introduction** chapter in the following link:  <http://docs.spring.io/spring-batch/trunk/reference/html/spring-batch-intro.html>  After reading, trainee should be able to answer these questions:   * What is a batch? Why do we need batch? * Provide some examples that we should implement them using batch. * Principles and guidelines that we should follow when implementing a batch? * Why we need Spring Batch while we can do everything manually? * If you have already known Quartz job, comparing to Spring batch job: how different they are?   Read **The Domain Language of Batch** chapter in the following link:  <http://docs.spring.io/spring-batch/trunk/reference/html/domain.html>  and **Configuring and Running a Job** chapter in the following link:  <http://docs.spring.io/spring-batch/trunk/reference/html/configureJob.html>  After reading, trainee should be able to answer these questions:   * What is Job? What are JobInstance, JobParameters, JobExecution and their relationships? * What is Step? StepExecution? * What is ExecutionContext? * What is JobRepository? * What is JobLaucher? * What are Item Reader, Item Writer and Item Processor? * Describe the full picture with all above things. * How to configure a job? How to run a job? | 2 |
|  | S | Setup your batch project | * Checkout the project from bitbucket * Import (Existing Projects into Workspace) project to IntelliJ. * Build then Run your project as Java Application (class **Application**). * You should see the following line in console: | 1 |
|  | R + E | Tasklet | Read section **TaskletStep** in the following link:  [http://docs.spring.io/spring-batch/reference/html/configureStep.html#taskletStep](http://docs.spring.io/spring-batch/reference/html/configureStep.html" \l "taskletStep)  Please note that the document is written using XML configuration but we are using Java configuration in our project.  You should read the xml configuration to understand what need to be defined then configure it using Java configuration (<http://docs.spring.io/spring-batch/reference/htmlsingle/#javaConfig>). For a comparison from XML to Java configuration, here is an example: <https://blog.codecentric.de/en/2013/06/spring-batch-2-2-javaconfig-part-1-a-comparison-to-xml/>  **Exercise 1:**  You are given a **sample-data.csv** that contains some lines. Each line is data of a Person.    Your job is implement a tasklet that read data from **sample-data.csv** then write to **output.csv** all the persons that have last name “Doe”.  The picture below shows our expected result: | 2 |
|  | R + E | Chunk-Oriented Processing | Read section **Chunk-Oriented Processing** in the following link:  [http://docs.spring.io/spring-batch/reference/html/configureStep.html#chunkOrientedProcessing](http://docs.spring.io/spring-batch/reference/html/configureStep.html" \l "chunkOrientedProcessing)  After reading, trainee should be able to answer these questions:   * What is chunk-oriented processing? * How to configure a chunk oriented processing step (reader-processor-writer)?   **Exercise 2:**  Redo *Exercise 1* but using chunk-oritented processing with Item Reader (+ Item Processor) and Item Writer instead of Tasklet.  Answer the following question:   * When to use tasklet and when to use chunk-oritented processing (with item reader-processor-writer)? * Why chunk-oriented processing is more flexible and advantage than tasklet? | 2 |
|  | R + E | IteamReader | Read following sections to have a deep look at various types of ItemReader and how to use them to read data from Database, from files.  **ItemReader** [http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#itemReader](http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html" \l "itemReader)  **Read data from File**   * FlatFileItemReader [http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#flatFileItemReader](http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html" \l "flatFileItemReader) * StaxEventItemReader (skim) <http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#StaxEventItemReader>   **Read data from Database**   * Cursor Based ItemReaders [http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#cursorBasedItemReaders](http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html" \l "cursorBasedItemReaders) * Paging ItemReaders [http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#pagingItemReaders](http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html" \l "pagingItemReaders)   Then read the **Step Scope** section in the following link, then answer the questions after:  [http://docs.spring.io/spring-batch/trunk/reference/html/configureStep.html#step-scope](http://docs.spring.io/spring-batch/trunk/reference/html/configureStep.html" \l "step-scope)   * How to tell Spring Batch that we don’t have anymore data to return from a reader? * What is step scope? * Why do we need step scope?   **Exercise 3:**   * Implement the reading part (ItemReader) in *Exercise 2* using FlatFileItemReader. * Now we don’t use the sample-data.csv provided in resources folder.  The file path will be retrieved from JobParameter.  Adapt your code to support this feature. Example:  (pathInputFile is retrieved from JobParameter)   Answer following questions:   * When is a bean marked with “Step” scope instantiated? * And when is it eligible to GC? | 4 |
|  | R + E | ItemProcessor | Read ItemProcessor section in the following link  [http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#itemProcessor](http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html" \l "itemProcessor)  After reading, trainee should be able to know:   * How to use ItemProcessor to enchane/validate data? * How to reject invalid data?   **Exercise 4:**  Adapt your processor to only accept person who meets these conditions:   * **PersonId** > 0 * Max length of **First name** is 20 * Max length of **Last name** is 20   Expected result: | 1 |
|  | R + E | ItemWriter | Read following sections to know how to use ItemWriter to write data to file, database.  **ItemWriter**  [http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#itemWriter](http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html" \l "itemWriter)  **Write data to file**   * FlatFileItemWriter [http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#flatFileItemWriter](http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html" \l "flatFileItemWriter) * StaxEventItemWriter (skim) <http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#StaxEventItemWriter>   **Write data to Database**   * Database ItemWriters [http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html#databaseItemWriters](http://docs.spring.io/spring-batch/trunk/reference/html/readersAndWriters.html" \l "databaseItemWriters) Please focus on how transaction is managed. * HibernateItemWriter <http://docs.spring.io/spring-batch/apidocs/org/springframework/batch/item/database/HibernateItemWriter.html> * JdbcBatchItemWriter <http://docs.spring.io/spring-batch/apidocs/org/springframework/batch/item/database/JdbcBatchItemWriter.html>   **Exercise 5a:**  Item Processor:   * Transform all the Persons who have last name start with “Super” to Engineer * EngineerId = PersonId * Name of engineer = First name + “ ” + Last name of Person. Max length of Name is 40 * Only accept Engineer who has EngineerId > 0   Item Writer  A **hsqldb** has been setup and embedded in your application. The DataSource is predefined and can be used in your bean with @Autowired    Script **spring-batch-training\src\main\resources\schema-all.sql** is run eachtime you launch your application. This script will clean up the database and create the needed tables.  Your job is to use **JdbcBatchItemWriter** to save all valid engineers to database. | 4 |
|  | R+E | Transaction in Spring Batch & skip policy | Read the following document: <https://blog.codecentric.de/en/2012/03/transactions-in-spring-batch-part-1-the-basics/>     * How the transaction is managed in Spring Batch?   **Exercise 5b:**  Now using [this file](https://svn.elca.ch/subversion/elcavn-newcomers/trunk/java/spring-batch/etc/sample-data-error.csv) as your input data. Answer the following questions:   * If I set chunk to **1**, how many engineers stored in the DB after the job completed? * If I set chunk to **2**, how many engineers stored in the DB after the job completed? * If I set chunk to **3**, how many engineers stored in the DB after the job completed? * If I set chunk to **5**, how many engineers stored in the DB after the job completed? * If I set chunk to **10**, how many engineers stored in the DB after the job completed? | 4 |
|  | C | Checkpoint | * Check exercises 1-5. * Check answers of trainee for all questions above. * Q&A meeting to verify understanding of trainee. | 1 |
|  | R + E | Controlling Step Flow | Read section **Controlling Step Flow** in the following link:  [http://docs.spring.io/spring-batch/reference/html/configureStep.html#controllingStepFlow](http://docs.spring.io/spring-batch/reference/html/configureStep.html" \l "controllingStepFlow)  **Exercise 6:**  Add 2 new Steps to your Job:   * CleanupStep: delete the input csv file that was processed. * MarkupStep: rename the input csv file to [original name].error when the job completed with error.   Configure your job to implement the following diagram:    Read the document: <http://docs.spring.io/spring-batch/reference/htmlsingle/#restartability> and <http://docs.spring.io/spring-batch/reference/html/configureStep.html> - chapter  Now, in case the job failed, try to restart it again. Compare between two options: restart the failed instance vs. start a new instance of the failed job. | 8 |
|  | R + E | Listener - Intercept Job/Step execution | Read sections in following links to know how to intercept Job/Step execution:  [http://docs.spring.io/spring-batch/reference/html/configureJob.html#interceptingJobExecution](http://docs.spring.io/spring-batch/reference/html/configureJob.html" \l "interceptingJobExecution)  [http://docs.spring.io/spring-batch/reference/html/configureStep.html#interceptingStepExecution](http://docs.spring.io/spring-batch/reference/html/configureStep.html" \l "interceptingStepExecution)  **Exercise 7:**  Add listeners to Job and Step execution to log following information:   * Error occurs inside a step / a job … * Total execution time of a Step * Total execution time of a Job | 2 |
|  | R + E | Unit Testing | Read following chapter to know how to write unit test for Spring Batch:  <http://docs.spring.io/spring-batch/reference/html/testing.html>  <http://www.mkyong.com/spring-batch/spring-batch-unit-test-example/>  **Exercise 8:**  Your project was already setup with Junit and Spring Batch Test.  The data source for test is configured in **BatchTestConfig**: it’s a memory-typed hsqldb which is gone after your application exits. Everytime a Test runs, **schema-hsqldb.sql** and **schema-data-table.sql** are executed so all needed tables are created.  Write 2 simple unit tests to test your chunkOrientedStep in class **ExampleSpringBatchTest**. Input data files are **sample-data.csv** and **sample-data-error.csv** in folder **src/test/resources**. | 4 |
|  | C | Checkpoint | * Coach check exercises 6-8 * Q&A meeting to verify understanding of trainee. | 1 |
|  | R + E | Advanced stuff | Read **Scaling and Parallel Processing** chapter in the following link:  <http://docs.spring.io/spring-batch/trunk/reference/html/scalability.html>  Read **Repeat** and **Retry** chapters in the following links:  <http://docs.spring.io/spring-batch/trunk/reference/html/repeat.html>  <http://docs.spring.io/spring-batch/trunk/reference/html/retry.html>  After reading, trainee should be able to know:   * How to use Multi-threaded Step? How to use Parallel Steps? * What is difference between Parallel and Multi-threaded step? When to use Parallel and when to use Multi-threaded step? * What is partitioning? How to do it? * What are important points we need to take care before doing parallel, multithread or partitioning processing?   **Exercise 9:**   * Back to the job created at step 7 – exercise 5a above, now when an engineer with EngineerId <=0 you throws a checked exception InvalidEngineerId to skip processing this Item. Then, you configure the job to skip rollback on this exception. You can refer the to link <http://docs.spring.io/spring-batch/reference/htmlsingle/#controllingRollback> * Try with the skip policy: only skip for first 5 items. If the 6th engineer with EngineerId <= 0 found then we don’t skip it. (try Google with keyword “faulttolerantstepbuilder example”) * Does skip policy affect the rollback of the items in the jobs? What is it used for?   **Exercise 10:**  Using [this data](https://svn.elca.ch/subversion/elcavn-newcomers/trunk/java/spring-batch/etc/sample-data-big-8mil.7z) as your input. It contains a csv file with 8 milions lines.  Keep the business of exercise 7. Adapt your batch job so it able to complete processing this big file in under 3 minutes. Implement your solution with 2 of 3 strategies below:   * Multi-threaded Step * Parallel Step * Partitioning   Compare your 2 implementations and your implementation in Exercise 7:  What are advantages and what we must pay for each implementation? | 8 |
|  | C | Expert validation | Validate the training result with expert:   * Expert’s visa: VLP * Validation date: * Remark: | 1 |

1. R = Reading, E = Exercise, C = Checkpoint, P = Presentation, S = Setup [↑](#footnote-ref-1)