Date: 21/04/2021
Spring Boot 9AM
Mr. RAGHU

Git Link:
https://github.com/javabyraghu/SpringBoot2BatchCsvToMongoDb

Spring Boot Batch : CSV to MongoDB Example

--Additional concepts--
a. Starter: Spring Data MongoDB

b. properties

```
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nit
```

c. ItemWriter : impl class is MongoItemWriter
  -> MongoTemplate (This is auto-configured object)
  -> collection name to create and insert data.

==============code================================

Name: SpringBoot2BatchCsvToMongoDBEx
Dep : Batch, Lombok, MongoDB, H2

1. Model class

```
package in.nareshit.raghu.model;

import lombok.Data;

@Data
public class Product {

	private Integer prodId;
	private String prodCode;
	private Double prodCost;
	private Double prodGst;
	private Double prodDiscount;
}
```

2. Processor class
3. Listener class

4. Batch Config

```
package in.nareshit.raghu.config;

import org.springframework.batch.core.Job;
import org.springframework.batch.core.JobExecution;
import org.springframework.batch.core.JobExecutionListener;
import org.springframework.batch.core.Step;
import
org.springframework.batch.core.configuration.annotation.EnableBatchPro
cessing;
import
org.springframework.batch.core.configuration.annotation.JobBuilderFact
ory;
```

```
import
org.springframework.batch.core.configuration.annotation.StepBuilderFac
tory;
import org.springframework.batch.core.launch.support.RunIdIncrementer;
import org.springframework.batch.item.ItemProcessor;
import org.springframework.batch.item.ItemReader;
import org.springframework.batch.item.ItemWriter;
import org.springframework.batch.item.data.MongoItemWriter;
import org.springframework.batch.item.file.FlatFileItemReader;
import
org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper;
import org.springframework.batch.item.file.mapping.DefaultLineMapper;
import
org.springframework.batch.item.file.transform.DelimitedLineTokenizer;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.data.mongodb.core.MongoTemplate;

import in.nareshit.raghu.model.Product;

@EnableBatchProcessing
@Configuration
public class BatchConfig {

        //1. reader object
        @Bean
        public ItemReader<Product> reader() {
                //JDK 1.7 Collections Type Inference
                FlatFileItemReader<Product> reader = new
FlatFileItemReader<>();
                reader.setResource(new
ClassPathResource("products.csv"));
                reader.setLineMapper(new DefaultLineMapper<>() {{
                        setLineTokenizer(new DelimitedLineTokenizer()
{{
                                setDelimiter(DELIMITER_COMMA);

setNames("prodId","prodCode","prodCost");
                        }});
                        setFieldSetMapper(new
BeanWrapperFieldSetMapper<>() {{
                                setTargetType(Product.class);
                        }});
                }});

                return reader;
        }
        //2. processor object
        @Bean
        public ItemProcessor<Product,Product> processor(){
                return (item)->{
                        item.setProdGst(item.getProdCost() * 0.12);
                        item.setProdDiscount(item.getProdCost() *
0.08);
                        return item;
                };
        }

        @Autowired
        private MongoTemplate template;

        //3. writer object
        @Bean
        public ItemWriter<Product> writer(){
                MongoItemWriter<Product> writer = new
MongoItemWriter<>();
                writer.setTemplate(template);
                writer.setCollection("products");
                return writer;
        }
        //4. listener object
        @Bean
        public JobExecutionListener listener(){
                return new JobExecutionListener() {
                        public void beforeJob(JobExecution je) {
                                System.out.println("STARTING
"+je.getStatus());
                        }
                        public void afterJob(JobExecution je) {
                                System.out.println("FINISHED
"+je.getStatus());
                        }
                };
        }
        //5. autowired SBF
        @Autowired
        private StepBuilderFactory sf;
        //6. Step object
        @Bean
        public Step stepA(){
                return sf.get("stepA")//name
                                .<Product,Product>chunk(3)//I,O,chunk
                                .reader(reader())
                                .processor(processor())
                                .writer(writer())
                                .build()
                                ;
        }
        //7. autowired JBF
        @Autowired
        private JobBuilderFactory jf;
        //8. Job object
        @Bean
        public Job jobA(){
                return jf.get("jobA")//name
                                .listener(listener())
                                .incrementer(new RunIdIncrementer())
                                .start(stepA())
                                .build();
        }
}
```

5. Runner class

```
package in.nareshit.raghu.runner;

import org.springframework.batch.core.Job;
import org.springframework.batch.core.JobParameters;
import org.springframework.batch.core.JobParametersBuilder;
import org.springframework.batch.core.launch.JobLauncher;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class MyJobRunner implements CommandLineRunner {

        @Autowired
        private JobLauncher launcher;
        @Autowired
        private Job jobA;


        public void run(String... args) throws Exception {
                JobParameters params = new JobParametersBuilder()
                                .addLong("time",
System.currentTimeMillis())
                                .toJobParameters();

                launcher.run(jobA, params);
        }

}
```

6. properties file

```
spring.batch.job.enabled=false
#spring.batch.initialize-schema=always

spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nit
```

7. producs.csv

```
10,PEN,200.0
11,BOOK,500.0
12,BOTTLE,600.0
13,MOBILE,1800.0
14,MOUSE,300.0
15,KEYBRD,900.0
16,BAG,600.0
```

Spring Boot Batch : MongoDB to CSV File

--Additional concepts--

a. Starter: Spring Data MongoDB

b. properties

```
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nit

c. ItemReader: impl class: MongoItemReader
  -> MongoTemplate
  -> collection name/Target Type
  -> Projection/Where conditon (Query)
  -> Sorting details
   ..etc
===============code=================================
Name: SpringBoot2BatchCsvToMongoDBEx
Dep : Batch, Lombok, MongoDB, H2

1. Model class
package in.nareshit.raghu.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class User {

        private Integer userId;
        private String userName;
        private String userRole;
        private String userDept;

}


2. Processor class
3. Listener class
4. Batch Config
package in.nareshit.raghu.config;


import java.util.HashMap;

import org.springframework.batch.core.Job;
import org.springframework.batch.core.JobExecution;
import org.springframework.batch.core.JobExecutionListener;
import org.springframework.batch.core.Step;
import
org.springframework.batch.core.configuration.annotation.EnableBatchPro
cessing;
import
org.springframework.batch.core.configuration.annotation.JobBuilderFact
ory;
import
org.springframework.batch.core.configuration.annotation.StepBuilderFac
tory;
import org.springframework.batch.core.launch.support.RunIdIncrementer;
import org.springframework.batch.item.ItemProcessor;
import org.springframework.batch.item.ItemReader;
import org.springframework.batch.item.ItemWriter;
import org.springframework.batch.item.data.MongoItemReader;
import org.springframework.batch.item.file.FlatFileItemWriter;
import
org.springframework.batch.item.file.transform.BeanWrapperFieldExtracto
r;
import
org.springframework.batch.item.file.transform.DelimitedLineAggregator;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.FileSystemResource;
import org.springframework.data.domain.Sort.Direction;
import org.springframework.data.mongodb.core.MongoTemplate;

import in.nareshit.raghu.model.User;

@EnableBatchProcessing
@Configuration
public class BatchConfig {

        @Autowired
        private MongoTemplate template;

        @Bean
        public ItemReader<User> reader(){
                MongoItemReader<User> reader = new MongoItemReader<>
();
                reader.setTemplate(template);
                reader.setTargetType(User.class);
                reader.setCollection("user");
                //reader.setQuery("{ uid: { $lt: 10} }");
                reader.setQuery("{ }");
                reader.setSort(new HashMap<String, Direction>() {{
                        put("_id", Direction.DESC);
                }});
                return reader;
        }

        @Bean
        public ItemProcessor<User,User> processor(){
                return item->item;
                //return new UserProcessor();
        }
        @Bean
        public ItemWriter<User> writer(){
                FlatFileItemWriter<User> writer = new
FlatFileItemWriter<>();
                writer.setResource(new
FileSystemResource("E:/myouts/usersmongodb.csv"));
                writer.setLineAggregator(new DelimitedLineAggregator<>
() {{
                        setDelimiter(",");
                        setFieldExtractor(new
BeanWrapperFieldExtractor<>() {{
                                setNames(new String[]
{"userId","userName","userRole","userDept"});
                        }});
                }});
                return writer;
        }
        @Bean
        public JobExecutionListener listener(){
                //return new MyJobListener();
                return new JobExecutionListener() {
                        public void beforeJob(JobExecution je) {
                                System.out.println(
                                                "Starting : "
+je.getStatus());
                        }
                        public void afterJob(JobExecution je) {
                                System.out.println(
                                                "Ending : "
+je.getStatus());
                        }
                };
        }

        @Autowired
        private StepBuilderFactory sf;

        @Bean
        public Step stepA(){
                return sf.get("stepA")
                                .<User,User>chunk(3)
                                .reader(reader())
                                .processor(processor())
                                .writer(writer())
                                .build();
        }
        @Autowired
        private JobBuilderFactory jf;

        @Bean
        public Job jobA(){
                return jf.get("jobA")
                                .listener(listener())
                                .incrementer(new RunIdIncrementer())
                                .start(stepA())
                                .build();
        }
}
```

5. Runner class

```
package in.nareshit.raghu.runner;

import org.springframework.batch.core.Job;
import org.springframework.batch.core.JobParametersBuilder;
import org.springframework.batch.core.launch.JobLauncher;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
```

```
@Component
public class MyJobRunner implements CommandLineRunner {

	@Autowired
	private JobLauncher launcher;
	@Autowired
	private Job jobA;

	public void run(String... args) throws Exception {
		launcher.run(jobA, new JobParametersBuilder()
				.addLong("time", 
System.currentTimeMillis())
				.toJobParameters());
		System.out.println("DONE");
	}
}
```

6. properties file

```
spring.batch.job.enabled=false
#spring.batch.initialize-schema=always

spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=nit
```

7. setup data in MongoDB

```
db.user.insert({"userId"  : 101,"userName": "ABCD","userRole": 
"ADMIN","userDept": "DEV"});
db.user.insert({"userId"  : 102,"userName": "MNO","userRole": 
"MGR","userDept": "QA"});
db.user.insert({"userId"  : 103,"userName": "AJAY","userRole": 
"SE","userDept": "DEV"});
db.user.insert({"userId"  : 104,"userName": "AHMED","userRole": 
"SEQ","userDept": "QA"});
db.user.insert({"userId"  : 105,"userName": "ANIL","userRole": 
"DER","userDept": "QA"});
```

==

Task:
1. CsvToMysql using JPA
2. MysqlToCsv using JPA
3. MySQLToXml***