Date : 20/02/2021
Spring Boot 6PM
Mr. RAGHU

Spring Boot Batch : CSV To MySQL

*) CSV File : (Command Separated Values File)
It holds data using symbol comma (,).

```
--order.csv--------------------------------
1016,OPEN-5869,500.256,HYD,VEN-CHN-85
1017,CLOSED-5069,500.256,HYD,VEN-CHN-85
1018,SHIP-5866,400.00,HYD,VEN-XYZ-50
1019,RETU-5098,800.256,HYD,VEN-XYZ-50
-----------------------------------------
```

*) Here, we use pre-defined ItemReader and ItemWriter classes.

Reader : FlatFileItemReader<T>

=> This reader is used to 'read data from files and Convert data into objects'

a. Read File name and location (Resource)
b. Read data Line by line (LineMapper)
c. Convert one Line values into variables data (LineTokenizer)
d. Create object using variable names (FieldSetMapper)

Writer : JdbcBatchItemWriter<T>

a. Create one Database Connection (DataSource)
b. Prepare one SQL query for INSERT using named params
(Data comes from objects)

```
  Ex:  INSERT INTO  PRODUCTS (pid,pcode,pcost)
       values (:prodId, :prodCode, :prodCost)
```

c. Read one Object data and create one INSERT SQL by replacing named params with object data.

```
 Ex: 1 Object --> 1 SQL
       INSERT INTO  PRODUCTS (pid,pcode,pcost)
       values (1019, "ABCD", 500.0)
```

d. Group Insert SQLs for chunk size and make n/w call for insert.

```
----batch config-----------------------------------------------------
------
package in.nareshit.raghu.config;
//ctrl+shift+O
@Configuration
@EnableBatchProcessing
public class BatchConfig {

        //a. Reader object
```

```
	@Bean
	public ItemReader<String> reader() {
		return new MyItemReader();
	}

	//b. processor object
	@Bean
	public ItemProcessor<String, String> processor() {
		return new MyItemProcesor();
	}
	//c. writer object
	@Bean
	public ItemWriter<String> writer() {
		return new MyItemWriter();
	}

	//d. listener object
	@Bean
	public JobExecutionListener listener() {
		return new MyJobListener();
	}

	//e. StepBuilderFactory Auto wire
	@Autowired
	private StepBuilderFactory sf;

	//f. Step object
	@Bean
	public Step stepA() {
		return sf.get("stepA")  //name
				.<String, String>chunk(3)
//<I,O>chunk(n)
				.reader(reader())  //reader object
				.processor(processor()) //processor
object
				.writer(writer())  //writer object
				.build() //create
				;
	}

	//g. JobBuilderFactory Auto wire
	@Autowired
	private JobBuilderFactory jf;

	//h. Job Object
	@Bean
	public Job jobA() {
		return jf.get("jobA")
				.incrementer(new RunIdIncrementer())
//calls steps
				.listener(listener()) //link listener
				.start(stepA())  //step1
				//.next(stepB()) //step2
				//.next(stepC())  //....
				.build()
				;
	}
```

```
}
```
