Date : 20/04/2021
Spring Boot 9AM
Mr. RAGHU

Spring Boot batch : MySQL To CSV

JdbcCursorItemReader<T> :
a. It needs database connection (DataSource)
b. Define ONE SELECT SQL query that gets data
   from DB into ResultSet
c. Use RowMapper<T> that converts
   data into Object Format

FlatFileItemWriter<T> :
a. Provide Resource (file+location)
b. Create one Line Data(Aggregate)
c. Provide Delimeter
d. Read data from Fields(object)

==========Database setup==========
1. create table

```
>  drop database boot9am;
> create database boot9am;
> use boot9am;
 create table usertab(uid int, uname varchar(20),
   urole varchar(20),udept varchar(20));
```

2. insert data

```
insert into usertab values(10,'A','ADMIN','DEV');
insert into usertab values(11,'B','ADMIN','QA');
insert into usertab values(12,'C','SE','DEV');
insert into usertab values(13,'D','TE','QA');
insert into usertab values(14,'E','ADMIN','BA');
insert into usertab values(15,'F','MG','BA');

>commit;
```

----coding order----------

1. create starter project

```
Name : springBoot2BatchMySQLToCsvEx
Dep  : Batch, Lombok, MySQL, JDBC API
```

2. application.proeprties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/boot9am
spring.datasource.username=root
spring.datasource.password=root

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

3. model class

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

import lombok.Data;
```

```
@Data
public class User {

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

}
```

4. Processor

```
package in.nareshit.raghu.processor;

import org.springframework.batch.item.ItemProcessor;

import in.nareshit.raghu.model.User;

public class UserProcessor
implements ItemProcessor<User, User>
{
	public User process(User item)
			throws Exception
	{
		return item;
	}
}
```

5. Listener

```
package in.nareshit.raghu.listener;

import org.springframework.batch.core.JobExecution;
import org.springframework.batch.core.JobExecutionListener;

public class MyJobListener implements JobExecutionListener {

	public void beforeJob(JobExecution je) {
		System.out.println("Starting : " +je.getStatus());
	}

	public void afterJob(JobExecution je) {
		System.out.println("Ending : " +je.getStatus());
	}

}
```

6. BatchConfig:-

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

import javax.sql.DataSource;

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.database.JdbcCursorItemReader;
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 in.nareshit.raghu.model.User;

@EnableBatchProcessing
@Configuration
public class BatchConfig {

        @Autowired
        private DataSource dataSource;

        @Bean
        public ItemReader<User> reader(){
                JdbcCursorItemReader<User> reader = new
JdbcCursorItemReader<>();
                reader.setDataSource(dataSource);
                reader.setSql("SELECT UID, UNAME, UROLE, UDEPT FROM
USERTAB");
                //reader.setRowMapper(new UserRowMapper());
                reader.setRowMapper(
                                (rs,n)->
                                new User(
                                                rs.getInt("uid")
                                                ,rs.getString("uname")

,rs.getString("urole"),
                                                rs.getString("udept")
                                                ));
                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/users.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();
	}
```

```
}

7. 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");
        }
}



=========RowMapper============================
package in.nareshit.raghu.mapper;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import in.nareshit.raghu.model.User;

public class UserRowMapper implements RowMapper<User> {

        public User mapRow(ResultSet rs, int n)
                        throws SQLException {
                User user = new User();
                user.setUserId(rs.getInt("uid"));
                user.setUserName(rs.getString("uname"));
                user.setUserRole(rs.getString("urole"));
                user.setUserDept(rs.getString("udept"));
                return user;
        }
}

(rs,n)->
new User(
        rs.getInt("uid")
```

```
        ,rs.getString("uname")
        ,rs.getString("urole"),
        rs.getString("udept")
));

*) BeanWrapperFieldExtractor
FieldExtractor : Read data from variable
 BeanWrapper: From Object given
                and create one line data with
                 "Variable,variable,Vairable,..." Format
=====Task=============================================
1. CSV File to MySQL DB
2. MySQL DB to CSV File
```