**SPRING-13 -Spring batch excel/csv to db**

Create a spring batch to fetch rows from an excel/csv file, convert the row’s values to uppercase and then write into the DB.

1. Create a Reader, processor and writer.

Reader will read the rows from excel DB, and since the spring batch reader uses only ItemReader, we have to convert the list of employees into itemreader.

1. Processor wil have the logic to process the records, here we have used processor to filter out the employees having dept as SALES.
2. We have used writer to write the employees into DB in the chunks of 10 employees.
3. We have configured this batch to run on application startup by using event listener.

**Code:**

**Job parameters:**

stepBuilderFactory.get("step1")  
 .<Employee, SalesEmployee>chunk(10)  
 .reader(excelUtil.excelItemReader(filePath))  
 .processor(processor)  
 .writer(writer)  
 .build();

**Run job on startup:**

@EventListener(ApplicationReadyEvent.class)  
public void run() throws Exception {  
 jobLauncher.run(  
 chunkDemo.job(),  
 new JobParametersBuilder().toJobParameters()  
 );  
}

**Reader logic to convert list to iterator:**

@StepScope  
public ItemReader<Employee> excelItemReader(String fileLocation) {  
 *log*.info("file location : " + fileLocation);  
 List<Employee> employeeList = new ArrayList<>();  
 try {  
 FileInputStream file = new FileInputStream(new File(fileLocation));  
 XSSFWorkbook workbook = new XSSFWorkbook(file);  
  
 Sheet sheet = workbook.getSheetAt(0);  
 int i = 0;  
 for (Row row : sheet) {  
 if (i == 0) {  
 assignCellValues(row);  
 i++;  
 continue;  
 }  
 addEmployeeToEmpList(employeeList, row);  
 i++;  
 }  
 file.close();  
 } catch (FileNotFoundException e) {  
 *log*.info("file not found at location : " + fileLocation);  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 var index = new AtomicInteger(0);  
 return () -> index.get() >= employeeList.size() ? null : employeeList.get(index.getAndIncrement());  
}

**Processor:**

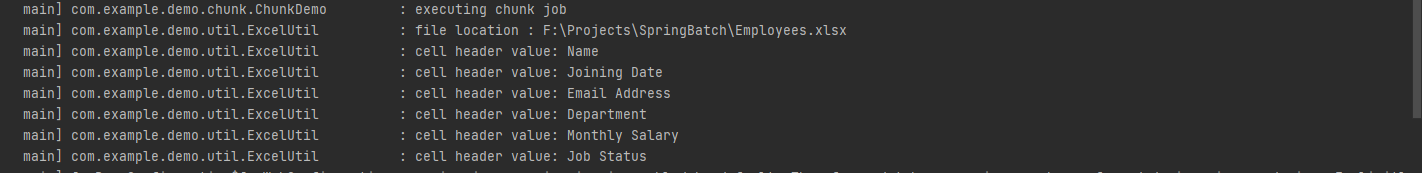
@Override  
public SalesEmployee process(Employee employee) throws Exception {  
 *log*.info("processing employee :: " + employee.getName() + ", " + employee.getEmail\_addr() + ", " + employee.getJoining\_date() + ", " + employee.getDepartment() + ", " + employee.getJob\_status() + ", " + employee.getMonthly\_salary());  
 if (employee.getDepartment().strip().equals("Sales")) {  
 *log*.info("employee of sales dept");  
 return new SalesEmployee()  
 .name(employee.getName())  
 .dept(employee.getDepartment())  
 .email\_addr(employee.getEmail\_addr())  
 .job\_status(employee.getJob\_status())  
 .joining\_date(employee.getJoining\_date())  
 .monthly\_salary(employee.getMonthly\_salary());  
 }  
 *log*.info("employee not of sales dept");  
 return null;  
}

**Writer:**

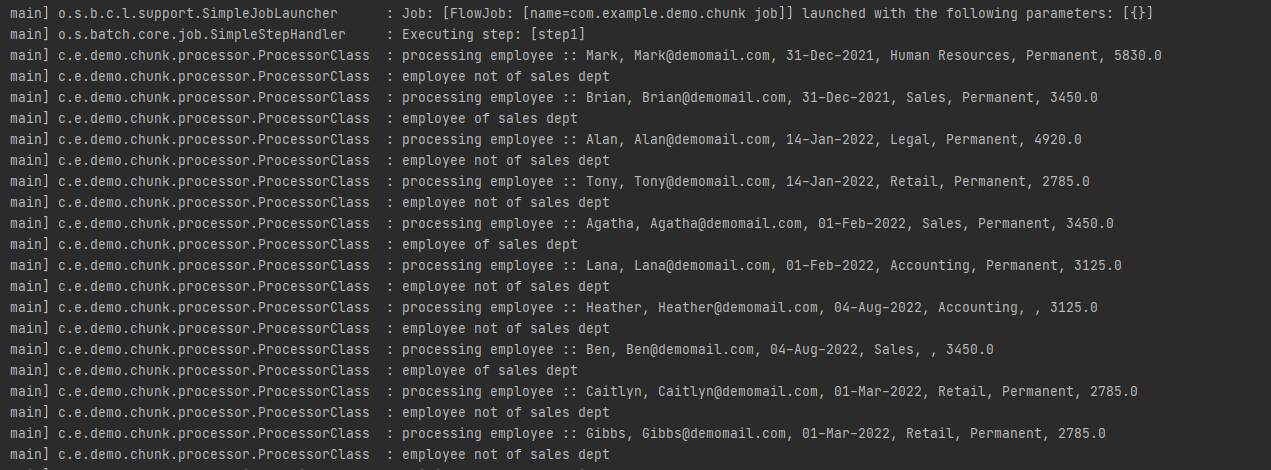
@Override  
public void write(List<? extends SalesEmployee> list) throws Exception {  
 *log*.info("writing employees to file");  
 salesRepository.saveAll(list);  
 *log*.info("writing done");  
}

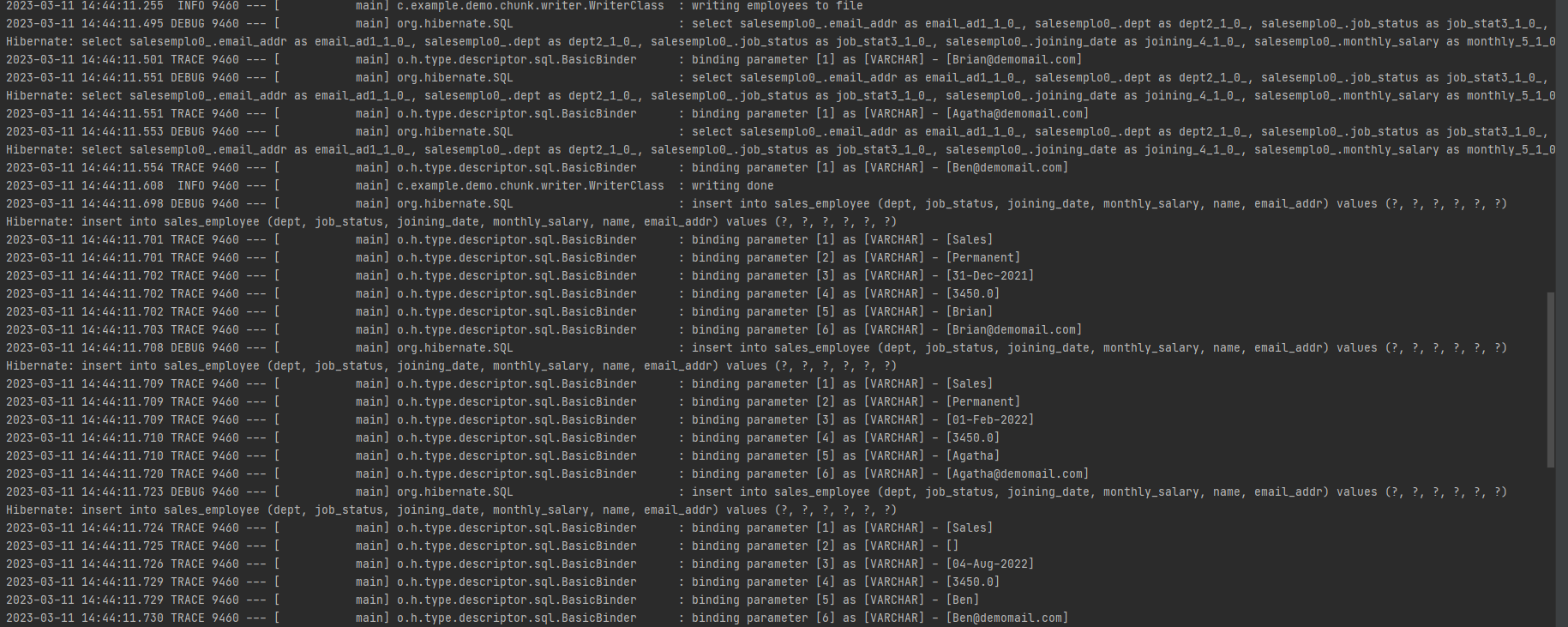
**SnapShots of run:**

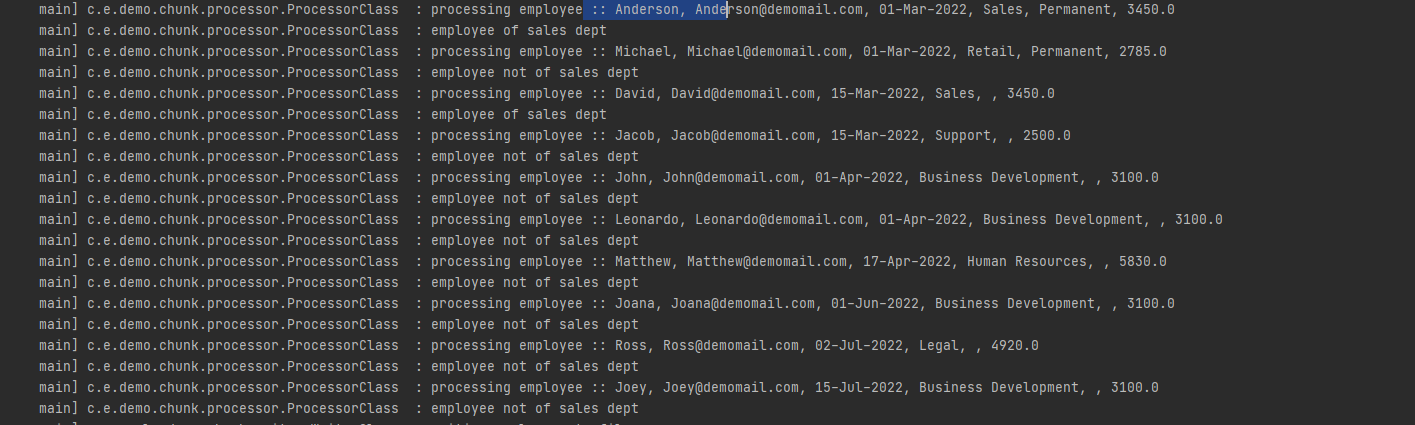
**Reader:**

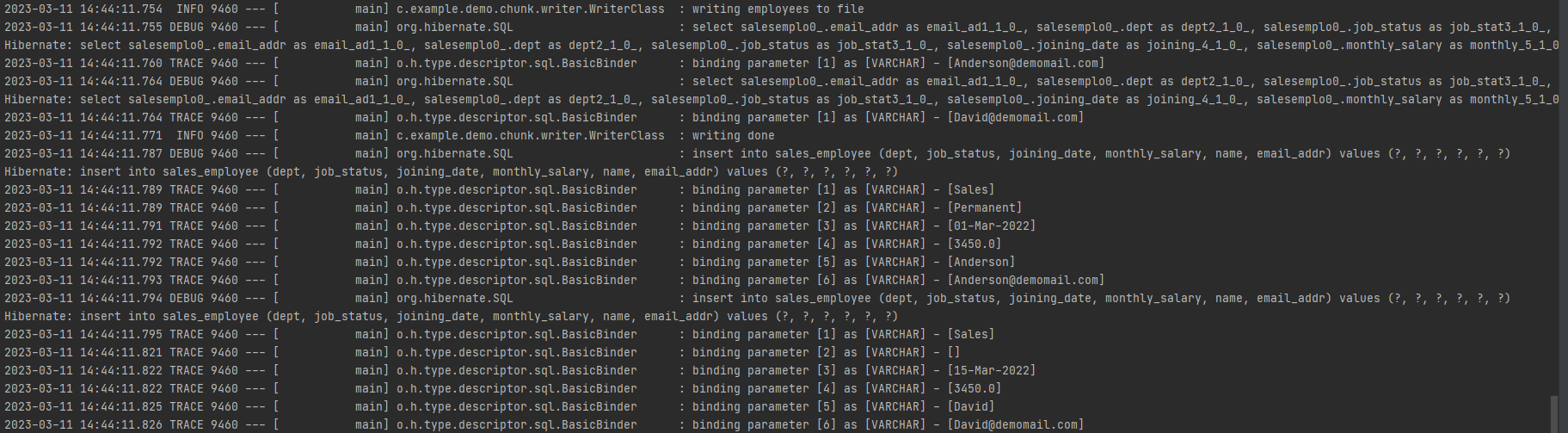


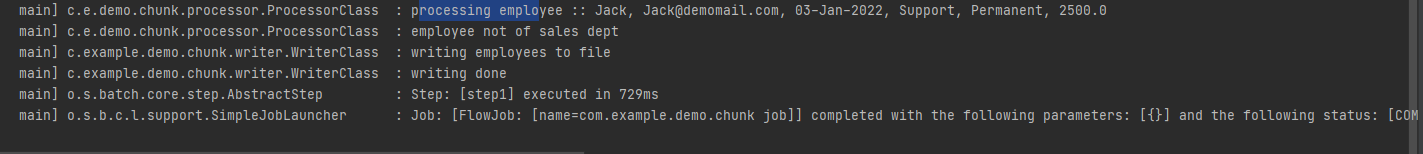
**Processor and writer in Chunks of 10**

1. 



1. 



1. 
2. **Rows written into DB:**

