POC (Proof of Cocept on Spring batch processing)

========

=>Custom ItemReader

=>Custom ItemWriter

=>Custom ItemProcessor => JobExecution Listener =>BatchConfig class

(w.r.t spring boot 2.x)

(enable Bathch Processing)

AutoConfiguration based @Autowired

-> JobBuilderFactory

note:: if the source repository is the DB s/w like oracle, MySQL and etc...

then there is no need of taking seperate H2 Logical DB for the Job Repository becoz the source DB s/w itself becomes the JobRepository keeping track of the all the activities..

-> Listener

-> StepBuilderFactory

@Autowiring

-> reader

->writer

->processor

-> @Bean method for Step obj creation

-> @Bean method for Job obj creation

=>Client App (Runner class)

Add spring -boot-starter-batch, h2 starter

->JobLauncher AutoConfiguration based @Autowired -> Job

@Autowiring

-> prepare JobParameters (optional)

->Run the job (launcher.run(job,parameters)

=======

=>h2 is InMemory DB s/w i.e DB will

be created in the JVM Memory of the RAM

and this DB will be used for JobRepository related

DB tables creation where the records will be maintained keeping track Job execution Activities

=> In spring boot 3.x all these are same but get

we do not JobBuilderFactory, StepBuilderFactory objs through AutoConfiguration, So we need to create the same objects using two param constructor of the JobBuilder, StepBuilder classes

=> Spring boot 3.x gives the following objects in batch processing through AutoConfiguration

a) JobLauncher b) Job Repository c) TransationManager

BatchApp1-POC [boot] >Spring Elements

#src/main/java

>com.nt

com.nt.config

> BatchConfig.java

com.nt.listener

>JobMonitoringListener.java

com.nt.processor

> BookDetailsProcessor.java

com.nt.reader

> BookDetailsReader.java

com.nt.runner

> BatchProcessing TestRunner.java

com.nt.writer

src/main/resources

application.properties

src/test/java

JRE System Library [JavaSE-11]

Maven Dependencies

>

>

>

>

src

>

target

w HELP.md

mvnw

mynw.cmd

Mpom.xml

//Listener

=========

package com.nt.listener;

import java.util.Date;

Using Spring boot 3.x

=========

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

@Component("jmListener")

```
public class JobMonitoringListener implements JobExecutionListener { private long startTime, endTime;
}
public JobMonitoringListener() {
System.out.println("JobMonitoringListener:: O-param constructor");
//Reader
package com.nt.reader;
import java.io.Serializable;
import org.springframework.batch.item.ItemReader;
import org.springframework.batch.item.Non TransientResourceException;
import org.springframework.batch.item.ParseException;
import org.springframework.batch.item.UnexpectedInputException; import
org.springframework.stereotype.Component;
@Component("bdReader")
public class BookDetails Reader implements ItemReader<String> { String books[]=new String[]
{"CRJ","TIJ","HFJ","EJ","BBJ"}; //Source int count=0;
public BookDetailsReader() {
System.out.println("BookDetails Reader:: O-param consturctor");
@Override
public void beforeJob(JobExecution jobExecution) { System.out.println("Job is about to beging at::"+new
Date());
}
startTime=System.currentTimeMillis();
System.out.println("Job Status ::"+jobExecution.getStatus());
@Override
public void afterJob(JobExecution jobExecution) {
System.out.println("Job completed at::"+new Date());
endTime=System.currentTimeMillis();
System.out.println("Job Status ::"+jobExecution.getStatus()); System.out.println("Job Exection time
::"+(endTime-startTime)); System.out.println("Job Exit Status ::"+jobExecution.getExitStatus());
}
}
}
}
@Override
public String read() throws Exception, UnexpectedInputException, ParseException,
NonTransientResourceException { System.out.println("BookDetailsReader.read()");
if(count<books.length) {
return books[count++];
```

```
}
else {
return null;
}
}
//Processor
==============
package com.nt.processor;
import org.springframework.batch.item.ItemProcessor;
import org.springframework.stereotype.Component;
@Component("bdProcessor")
public class BookDetailsProcessor implements Item Processor<String, String> {
public BookDetailsProcessor() {
}
System.out.println("BookDetails Processor:: O-param constructor");
//writer
package com.nt.writer;
import java.util.List;
import org.springframework.batch.item.ItemWriter; import org.springframework.stereotype.Component;
@Component("bdWriter")
public class BookDetailsWriter implements ItemWriter<String> {
@Override
Chunk
public void write <? extends String> items) throws Exception {
System.out.println("BookDetailsWriter.write()");
items.forEach(System.out::println);
@Override
public String process(String item) throws Exception {
System.out.println("BookDetailsProcessor.process()");
String bookWith Title=null;
if(item.equalsIgnoreCase("CRJ"))
bookWith Title=item+" by HS and PN";
else if(item.equalsIgnoreCase("TIJ")) bookWithTitle=item+" by BE"; else if(item.equalsIgnoreCase("HFJ"))
bookWithTitle=item+" by KS"; else if(item.equalsIgnoreCase("EJ")) bookWith Title=item+" by JB"; else
if(item.equalsIgnoreCase("BBJ")) bookWithTitle=item+" by RNR";
return bookWith Title;
```

In spring boot 3.x

===============

}

}

=> StepBuilderFactory is deprecated alternate is StepBuilder => JobBuilderFactory is deprecated alternate is JobBuilder => chunk(size) is dreprecated alternate chunk(size, txMgmr)

}

}

BatchConfig.java

=======

```
package com.nt.config;

import org.springframework.batch.core.Job;

import org.springframework.batch.core.Step;

import org.springframework.batch.core.job.builder.JobBuilder;

import org.springframework.batch.core.launch.support.Runidincrementer; import
org.springframework.batch.core.repository.JobRepository; import
org.springframework.batch.core.step.builder.StepBuilder; import
org.springframework.beans.factory.annotation.Autowired; import
org.springframework.context.annotation.Bean; import
org.springframework.context.annotation.Configuration; import
org.springframework.transaction.PlatformTransactionManager;

import com.nt.listener.JobMonitoringListener; import com.nt.processor.BookItemProcessor;

import com.nt.reader.BookItemReader; import com.nt.writer.BookItemWriter;

@Configuration

public class BatchConfig {

@Autowired

private BookItemReader reader;

}

@Autowired

private BookItemWriter writer;

@Autowired

private BookItemProcessor processor; @Autowired

private JobMonitoringListener listener;

@Bean(name="step1")

public Step createStep1(Job Repository repository,

Platform Transaction Manager txMgmr) {

System.out.println("BatchConfig.createStep1()");

return new StepBuilder("step1", repository)

.<String, String>chunk(2, txMgmr)

.reader(reader)

.processor(processor)
.writer(writer)
}
.build();
@Bean(name="job1")
public Job createJob1(Job Repository repository,Step step1) {
System.out.println("BatchConfig.createJob1()");
return new JobBuilder("job1", repository)
.listener(listener)
.incrementer(new RunIdincrementer())
.start(step1)
.build();
}
```

Runner class.,-

```
package com.nt.runner;
import java.util.Random;
import org.springframework.batch.core.Job;
import org.springframework.batch.core.JobExecution; 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 Batch ProcessingTestRunner implements CommandLineRunner {
@Autowired
private JobLauncher launcher;
@Autowired
private Job job;
@Override
public void run(String... args) throws Exception {
//prepare Job Parameters
JobParameters params=new Job ParametersBuilder()
//run the job
.addLong("time",System.currentTimeMillis()).toJobParameters();
JobExecution exeution-launcher.run(job, params);
/*System.out.println("Job execution status ::"+exeution.getStatus());
System.out.println("Exit Status ::"+exeution.getExitStatus());
System.out.println(" Job Id"+exeution.getJobId());*/
}
```

}

note:: JobBuildFactory, StepBuilderFactory objs are not coming as the spring bean through AutoConfiguration process in spring boot 3.x i.e they are coming only in spring boot 2.x

//application.properties

spring.batch.job.enabled=false

# Indicates wheate batch code should execute

spring.batch.jdbc.initialize-schema-always

It use underlying Db s/w to

create lots db tables to track

of job execution related operations..

batch_job_execution

batch_step_execution_seq

batch_step_execution_context

‣ batch_step_execution

batch job_seq

Datch_job_instance

hatch ich everution_seq

Datch_job_ batch job_execution_params batch_job_execution_context

batch_job_execution

batch_job_execution_params

batch job_instance

batch_job_seq

batch_step_execution

batch_step_execution_seq

on the app startup or on demand

(true

(false :: on demand when

(default)

:: on the app startup)

lanucher.run(-) is called)

possible values ::

never, always embedded

Another way of writing BatchConfig class

===========

package com.nt.config;

import org.springframework.batch.core.Job;

import org.springframework.batch.core.Step;

import org.springframework.batch.core.configuration.annotation.EnableBatch Processing; import

```
org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import
org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import
org.springframework.batch.core.launch.support.RunIdIncrementer; import
org.springframework.beans.factory.annotation.Autowired; import
org.springframework.context.annotation.Bean; import
org.springframework.context.annotation.Configuration;
import com.nt.listener.JobMonitoringListener;
import com.nt.processor.BookDetailsProcessor;
import com.nt.reader.BookDetailsReader; import com.nt.writer.BookDetailsWriter;
@Configuration
public class BatchConfig1 {
@Bean
public JobMonitoringListener createListener() {
return new JobMonitoringListener();
```

Can i configure use-defined classes as spring beans using @Bean methods?

Ans) Possible, but not recommanded

```
}
@Bean
public BookDetailsWriter createWriter() {
return new BookDetailsWriter();
}
@Bean
public BookDetails Processor createProcessor() {
return new BookDetails Processor();
}
@Bean
public BookDetails Reader createReader() {
return new BookDetails Reader();
}
@Bean(name="step1")
public Step createStep1(Job Repository repository,
PlatformTransaction Manager txMgmr) {
System.out.println("BatchConfig.createStep1()");
return new StepBuilder("step1", repository)
.<String,String>chunk(2, txMgmr)
.reader(createReader())
.processor(createProcessor())
.writer(createWriter())
```

}

.build();

@Bean(name="job1")

public Job createJob1(JobRepository repository,Step step1) {

System.out.println("BatchConfig.createJob1()");

return new JobBuilder("job1", repository)

.listener(createListener())

.incrementer(new RunId Incrementer()) .start(step1)

}

}

=> Types of inner classes in java

=> types of blocks java class

.build();

=> what is difference b/w instance block and static block

=> When we have constructor to write initialziation logic, then when do we need

instance block?

do

To see Job Repository that is created in the H2 DB, we need to following operations

step1) add spring boot starter web

to the pom.xml file

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-web --> <dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

step2) add the following properties in the application.properites file

# H2 DB port

server.port=6061 (Tomcat server port and h2 console page)

#h2 DB properties

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.username=root

spring.datasource.password=root

spring.h2.console.enabled=true

JDBC properties of the H2DB

We need to add this

becoz the h2 console is given

as the web page in

the web application

step3) perform the following operations

a) run the application normally and make sure that the application is in running mode until we stop it

(i) add the following code in the Runner class

b) open h2 DB console page

System.out.println("Press any key to continue");

System.in.read();

http://localhost:6061/h2-console

←

localhost:6061/h2-console/login.jsp?jsessionid=16ecc787dbe6d9fd10907e8811bbf840

English

Preferences Tools Help

Login

Saved Settings: Setting Name:

Generic H2 (Embedded)

Generic H2 (Embedded)

Save Remove

Driver Class:

org.h2.Driver

JDBC URL:

jdbc:h2:mem:testdb

User Name:

root

Password:

root

Connect Test Connection

BatchConfig,java (In spring boot 2.x)

=====

package com.nt.config;

import org.springframework.batch.core.Job;

import org.springframework.batch.core.Step;

import org.springframework.batch.core.configuration.annotation.Enable Batch Processing; import org.springframework.batch.core.configuration.annotation.JobBuilderFactory; import org.springframework.batch.core.configuration.annotation.StepBuilderFactory; import org.springframework.batch.core.launch.support.Runidincrementer;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration;

```
import org.springframework.context.annotation.EnableAspectJAutoProxy;
import com.nt.listener.JobMonitoringListener;
import com.nt.processor.BookItemProcessor; import com.nt.reader. BookItemReader; import
com.nt.writer.BookItemWriter;
@Configuration
@EnableBatch Processing
public class BatchConfig {
@Autowired
private JobBuilderFactory jobFactory;
@Autowired
private StepBuilderFactory stepFactory;
@Autowired
private BookItemReader reader;
@Autowired
private BookItemWriter writer;
@Autowired
private BookItemProcessor processor;
@Autowired
private JobMonitoringListener listener;
@Bean(name="step1")
public Step createStep1() {
System.out.println("BatchConfig.createStep1()");
return stepFactory.get("step1")
.<String, String>chunk(2) .reader(reader) .processor(processor) .writer(writer)
.build();
}
@Bean(name="job1")
public Job createJob1() {
System.out.println("BatchConfig.createJob1()");
return jobFactory.get("job1")
.listener(listener)
.incrementer(new RunIdIncrementer())
.start(createStep1())
.build();
}
```

jdbc:h2:mem:testdb

BATCH_JOB_EXECUTION

+

BATCH_JOB_EXECUTION_CONTEXT

+

BATCH_JOB_EXECUTION_PARAMS

+

BATCH_JOB_INSTANCE

+

BATCH_STEP_EXECUTION

+

BATCH_STEP_EXECUTION_CONTEXT

+

INFORMATION_SCHEMA

=>The following DB tables are created representing JobRepository activities