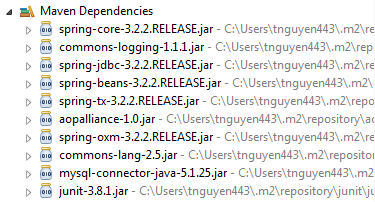
Hôm nay tìm hiểu cái vụ Batch, k hiểu k về.

# Ví dụ trên trang mkyong.com

<https://www.mkyong.com/spring-batch/spring-batch-hello-world-example/>

Trước khi có Spring-Batch và Spring-Batch unit test:



Sử dụng Dependency sau:

<!-- Spring Batch dependencies -->

<dependency>

<groupId>org.springframework.batch</groupId>

<artifactId>spring-batch-core</artifactId>

<version>${spring.batch.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.batch</groupId>

<artifactId>spring-batch-infrastructure</artifactId>

<version>${spring.batch.version}</version>

</dependency>

<!-- Spring Batch unit test -->

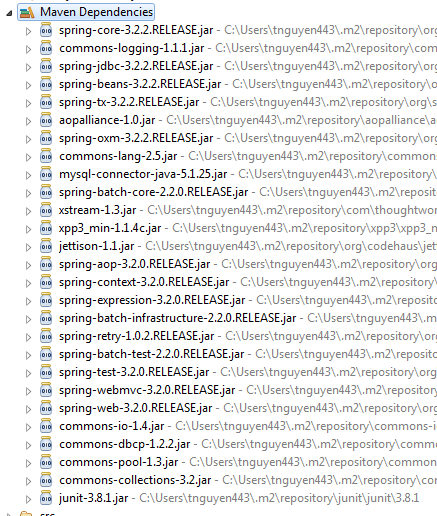
<dependency>

<groupId>org.springframework.batch</groupId>

<artifactId>spring-batch-test</artifactId>

<version>${spring.batch.version}</version>

</dependency>

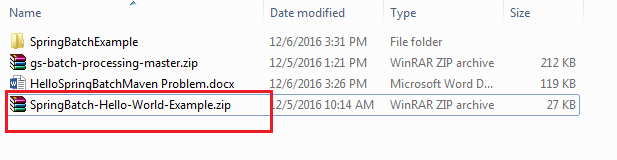


Project này cần MySQL, cài xampp xong thì chạy được.

## Step 1 – Read CSV files from folder A, process, write it to folder B. “READ-PROCESS-WRITE”

Map nó qua cái projectHelloSpringMaven mãi không được. Không hiểu tại sao.

Chán quá export cái project này ra là chạy thôi. Kệ bà nó đi.



Muốn ghi ra thêm file csv thì sửa file config job lại như sau:

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:batch=*"http://www.springframework.org/schema/batch"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/batch*

*http://www.springframework.org/schema/batch/spring-batch-2.2.xsd*

*http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.2.xsd*

*"*>

<import resource=*"../config/context.xml"* />

<import resource=*"../config/database.xml"* />

<bean id=*"report"* class=*"com.mkyong.model.Report"* scope=*"prototype"* />

<bean id=*"itemProcessor"* class=*"com.mkyong.CustomItemProcessor"* />

<batch:job id=*"helloWorldJob"*>

<batch:step id=*"step1"* next=*"step2"*>

<batch:tasklet>

<batch:chunk reader=*"cvsFileItemReader"* writer=*"xmlItemWriter"*

processor=*"itemProcessor"* commit-interval=*"10"*>

</batch:chunk>

</batch:tasklet>

</batch:step>

<batch:step id=*"step2"*>

<batch:tasklet>

<batch:chunk reader=*"cvsFileItemReader"* writer=*"cvsItemWriter"*

processor=*"itemProcessor"* commit-interval=*"10"*>

</batch:chunk>

</batch:tasklet>

</batch:step>

</batch:job>

<bean id=*"cvsFileItemReader"* class=*"org.springframework.batch.item.file.FlatFileItemReader"*>

<property name=*"resource"* value=*"classpath:cvs/input/report.csv"* />

<property name=*"lineMapper"*>

<bean class=*"org.springframework.batch.item.file.mapping.DefaultLineMapper"*>

<property name=*"lineTokenizer"*>

<bean

class=*"org.springframework.batch.item.file.transform.DelimitedLineTokenizer"*>

<property name=*"names"* value=*"id,sales,qty,staffName,date"* />

</bean>

</property>

<property name=*"fieldSetMapper"*>

<bean class=*"com.mkyong.ReportFieldSetMapper"* />

<!-- if no data type conversion, use BeanWrapperFieldSetMapper to map

by name <bean class="org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper">

<property name="prototypeBeanName" value="report" /> </bean> -->

</property>

</bean>

</property>

</bean>

<bean id=*"xmlItemWriter"* class=*"org.springframework.batch.item.xml.StaxEventItemWriter"*>

<property name=*"resource"* value=*"file:xml/outputs/report.xml"* />

<property name=*"marshaller"* ref=*"reportMarshaller"* />

<property name=*"rootTagName"* value=*"report"* />

</bean>

<bean id=*"cvsItemWriter"* class=*"org.springframework.batch.item.file.FlatFileItemWriter"*>

<!-- write to this csv file -->

<property name=*"resource"* value=*"file:xml/outputs/report.csv"* />

<property name=*"shouldDeleteIfExists"* value=*"true"* />

<property name=*"lineAggregator"*>

<bean

class=*"org.springframework.batch.item.file.transform.DelimitedLineAggregator"*>

<property name=*"delimiter"* value=*","* />

<property name=*"fieldExtractor"*>

<bean

class=*"org.springframework.batch.item.file.transform.BeanWrapperFieldExtractor"*>

<property name=*"names"* value=*"id, sales, qty, staffName, date"* />

</bean>

</property>

</bean>

</property>

</bean>

<bean id=*"reportMarshaller"* class=*"org.springframework.oxm.jaxb.Jaxb2Marshaller"*>

<property name=*"classesToBeBound"*>

<list>

<value>com.mkyong.model.Report</value>

</list>

</property>

</bean>

</beans>

# 2.

<https://spring.io/guides/gs/batch-processing/>

# 3. Các khái niệm

Batch hoặc Batch processing là từ được sử dụng trong ngành công nghệ thông tin để tả cách xử lý hàng loạt của những chương trình vi tính mà không cần sự can thiệp của nhân viên.

Ví dụ: Mỗi cuối tháng, các chương trình tính lương tính và in hàng nghìn bản lương để trả cho nhân viên. Các chương trình này thường chạy theo cách batch:Đọc tập tin "hồ sơ nhân viên" của hãng

Tính lương tháng cho nhân viên

In bản lương tháng cho nhân viên

Đọc và xử lý hồ sơ tiếp theo...

Spring Batch is một framework cho việc xử lý batch-một chuỗi các công việc,trong spring batch,một công việc bao gồm nhiều steps và mỗi step bao gồm một nhiệm vụ là READ-PROCESS-WRITE hoặc một task chỉ gồm một hành động(tasklet)

Đối với việc “READ-PROCESS-WRITE”,nó có nghĩa là đọc dữ liệu từ một nguồn nào đó(csv,xml hoặc database),xử lý và ghi vào một nguồn khác như(csv,xml và database).Ví dụ một bước đọc dữ liệu từ một file csv,xử lý dữ liệu và ghi nó vào database,Spring Batch cung cấp nhiều các class để đọc và ghi csv,xml và database. Đối với nhiệm vụ mà có một hành động như là dọn dẹp các nguyên trước hoặc sau khi một step bắt đầu hoặc kết thúc

Các step có thể được xâu chuỗi cùng nhau để chạy như một job

1 Job = Mnay steps

1 step = 1 READ-PROCESS-WRITE or 1 Tasklet.

Job = {Step 1 -> Step 2 -> Step 3} (Chained together)

Spring Batch cung cấp một tập hợp các reusable function (các hàm có thể tái sử dụng). Các hàm này giữ vai trò trọng yếu khi xử lý một lượng lớn các bản ghi trong CSDL.Bao gồm logging/tracing,quản lý transaction ,thống kê xử lý job,khởi động lại job,skip và quản lý tài nguyên.Nó cũng cung cấp nhiều các dịch vụ tiện ích và các tính năng có thể ở quy mô lớn,và nâng cao hiệu năng của các batch job thông qua việc tối ưu hóa và công nghệ partitioning .Đơn gian cũng như phức tạp,các công việc xử lý batch lớn có thể tận dụng framework này trong việc quản lý khả năng mở rộng để xử lý khối lượng lớn thông tin

# 4. Crontab Pattern

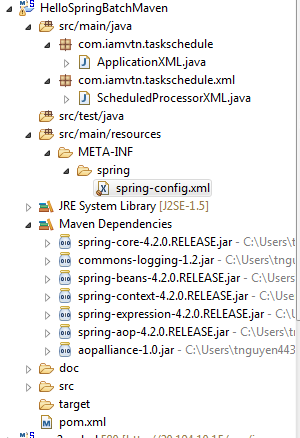
“0 0 \* \* \* \*” = Mỗi đầu giờ chạy một lần.  
“\*/10 \* \* \* \* \*” = Cứ 10 giây chạy một phát.  
“0 0 8-10 \* \* \*” = Chạy liên tục cứ đầu giờ từ 8 giờ đến 10 giờ trong ngày  
“0 0/30 8-10 \* \* \*” = Cứ 30 phút chạy một lần liên tục từ lúc 8 giờ đến 10 giờ (sẽ chạy lúc 8:00, 8:30, 9:00, 9:30 và 10 giờ job sẽ chạy vào mọi người trong tuần)  
“0 0 9-17 \* \* MON-FRI” = Chạy liên tục vào đầu giờ từ lúc 9h sáng đến 17h chiều, vào thứ 2 đến thứ 6 (Monday to Friday)  
“0 0 0 25 12 ?” = Luôn luôn chạy vào lúc nửa đêm giáng sinh vào mọi năm.

Trang để thiết lập crontab pattern: <http://www.htmlbasix.com/crontab.shtml>

# 5. Sử dụng Spring batch xài crontab pattern để chạy chương trình trong 1 thời điểm cố định

<https://namvuhn.wordpress.com/2015/08/06/tips-tao-scheduling-tasks-voi-spring-task/>

## Cấu trúc



## pom.xml

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>org.springframework</groupId>

<artifactId>HelloSpringBatchMaven</artifactId>

<version>0.1.0</version>

<properties>

<java.version>1.8</java.version>

<spring-version>4.2.0.RELEASE</spring-version>

</properties>

<dependencies>

<!-- Spring framework -->

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring-version}</version>

</dependency>

<!-- Spring Beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring-version}</version>

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring-version}</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring-version}</version>

</dependency>

</dependencies>

</project>

## spring-config.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:task=*"http://www.springframework.org/schema/task"*

xsi:schemaLocation=*" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.2.xsd http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task-3.2.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd"*>

<context:annotation-config />

<context:component-scan base-package=*"com.iamvtn.taskschedule.xml"* />

<task:scheduled-tasks>

<task:scheduled ref=*"scheduledProcessorXML"* method=*"process"*

cron=*"0 0/1 10-12 \* \* \*"* />

</task:scheduled-tasks>

</beans>

Bạn có thể thấy ở đây tôi dùng thẻ “**context:annotation-config**” và “**context:component-scan**” với tham số “**base-package**” là “**com.iamvtn.taskschedule.xml**“, có nghĩa là spring sẽ tự động quét tất cả các class có trong package trên (bao gồm cả các class trong package con của nó) rồi tự động khởi tạo các đối tượng. Việc này cũng tương tự như các bạn thẻ “bean” để định nghĩa việc khởi tạo từng đối tượng một bằng tay. Một bên là “manual” một bên là “auto” không khác nhau là mấy

Còn thẻ “**task:scheduled-tasks”** để chứa các “cron job”, ở đây tôi chỉ có 1 cron job link đến class là ScheduledProcessorXML method là process với tham số **cron=”0 35 10 \* \* MON-FRI”** nó có nghĩa là cứ đúng **10 giờ 35 phút sáng** các ngày từ **thứ 2 đến thứ 6**, hệ thống sẽ tự động gọi đến hàm **process** trong lớp **ScheduledProcessorXML (Cái này là ví dụ trên mạng, mình đã sửa lại).**

## ScheduledProcessorXML

**package** com.iamvtn.taskschedule.xml;

**import** java.time.LocalDateTime;

**import** org.springframework.stereotype.Service;

@Service

**public** **class** ScheduledProcessorXML {

**public** **void** process() {

System.***out***.println("Processing at " + LocalDateTime.*now*() + " ntthuat-test");

}

}

## ApplicationXML

**package** com.iamvtn.taskschedule;

**import** org.springframework.context.support.FileSystemXmlApplicationContext;

**public** **class** ApplicationXML {

@SuppressWarnings("resource")

**public** **static** **void** main(String[] args) {

System.***out***.println("Initializing...");

**new** FileSystemXmlApplicationContext(**new** String[] { "classpath:META-INF/spring/spring-config.xml" }).start();

System.***out***.println("Initialized.");

}

}

## Result:

