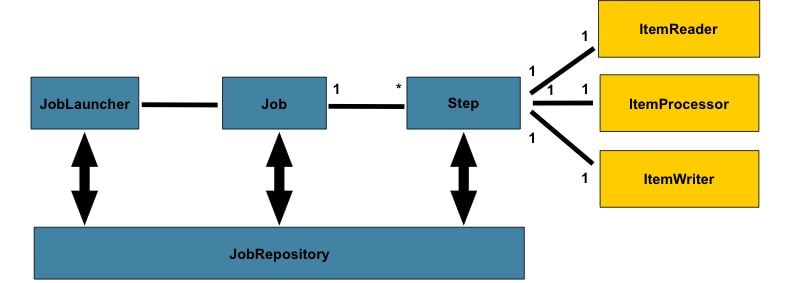
**Spring Batch**

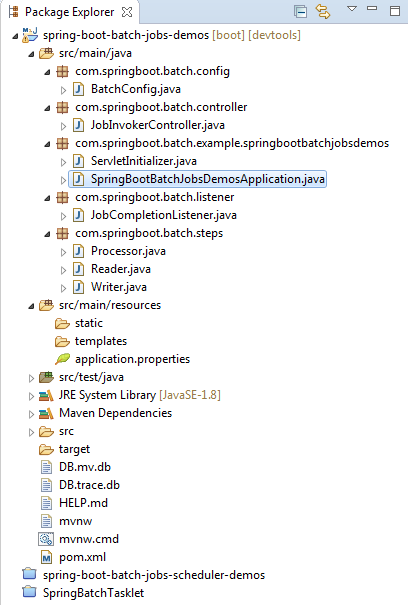
**How Spring Batch works?**



* **step -**A Step that delegates to a Job to do its work. This is a great tool for managing dependencies between jobs, and also to modularize complex step logic into something that is testable in isolation. The job is executed with parameters that can be extracted from the step execution, hence this step can also be usefully used as the worker in a parallel or partitioned execution.
* **ItemReader -**Strategy interface for providing the data. Implementations are expected to be stateful and will be called multiple times for each batch, with each call to read() returning a different value and finally returning null when all input data is exhausted. Implementations need not be thread-safe and clients of a ItemReader need to be aware that this is the case. A richer interface (e.g. with a look ahead or peek) is not feasible because we need to support transactions in an asynchronous batch.
* **ItemProcessor -**Interface for item transformation. Given an item as input, this interface provides an extension point which allows for the application of business logic in an item oriented processing scenario. It should be noted that while it's possible to return a different type than the one provided, it's not strictly necessary. Furthermore, returning null indicates that the item should not be continued to be processed.
* **ItemStreamWriter -**Basic interface for generic output operations. Class implementing this interface will be responsible for serializing objects as necessary. Generally, it is responsibility of implementing class to decide which technology to use for mapping and how it should be configured. The write method is responsible for making sure that any internal buffers are flushed. If a transaction is active it will also usually be necessary to discard the output on a subsequent rollback. The resource to which the writer is sending data should normally be able to handle this itself.

**Spring Boot Batch Job Simple Example**

In this example we develop a simple Spring Boot Batch application.  
Consider an environment where users have to do a lot of batch processing. This will be quite different from a typical web application which has to work 24/7. But in classic environments it's not unusual to do the heavy lifting for example during the night when there are no regular users using your system. Batch processing includes typical tasks like reading and writing to files, transforming data, reading from or writing to databases, creates reports, import and export data and things like that. Often these steps have to be chained together or you have to create more complex workflows where you have to define which job steps can be run in parallel or have to be run sequentially etc. That's where a framework like Spring Batch can be very handy.   
Spring Boot Batch provides reusable functions that are essential in processing large volumes of records, including logging/tracing, transaction management, and job processing statistics, job restart, skip, and resource management. It also provides more advanced technical services and features that will enable extremely high-volume and high performance batch jobs though optimization and partitioning techniques. Simple as well as complex, high-volume batch jobs can leverage the framework in a highly scalable manner to process significant volumes of information.



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

<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>

<parent>

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

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

<version>2.1.6.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.springboot.batch.example</groupId>

<artifactId>spring-boot-batch-jobs-demos</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>spring-boot-batch-jobs-demos</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>provided</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

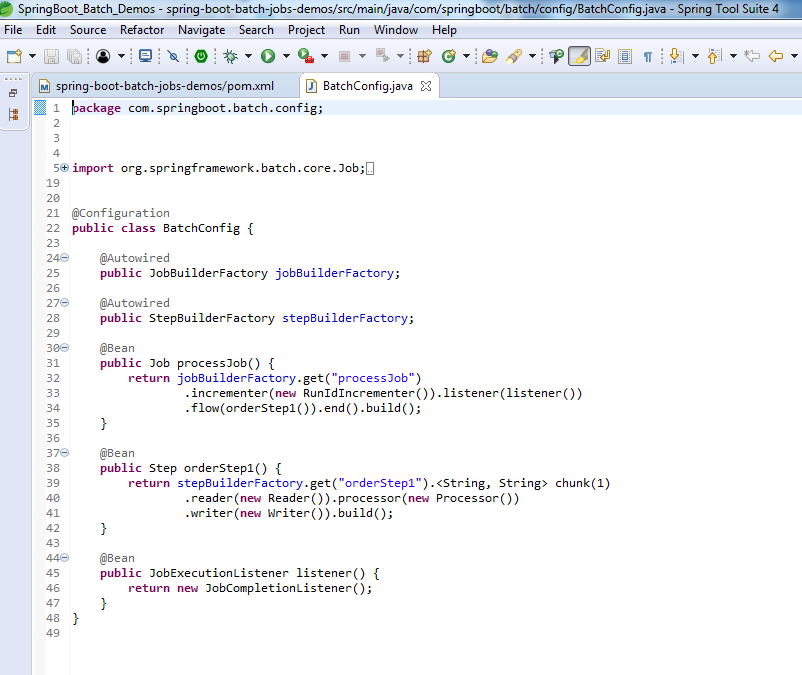
<artifactId>spring-boot-maven-plugin</artifactId>

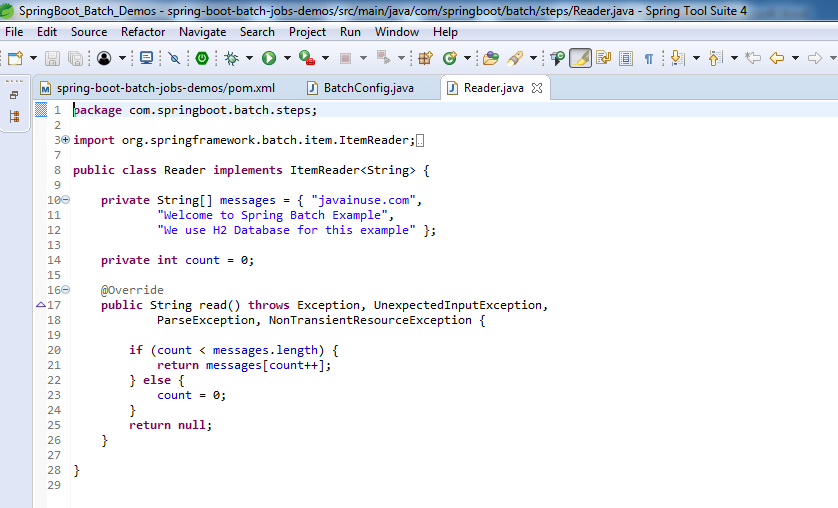
</plugin>

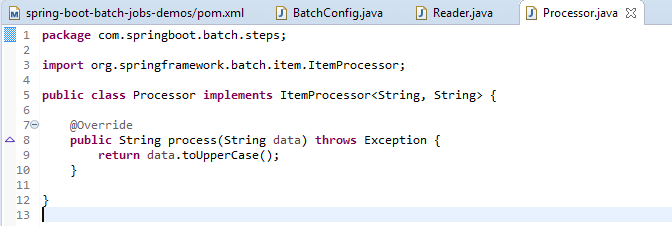
</plugins>

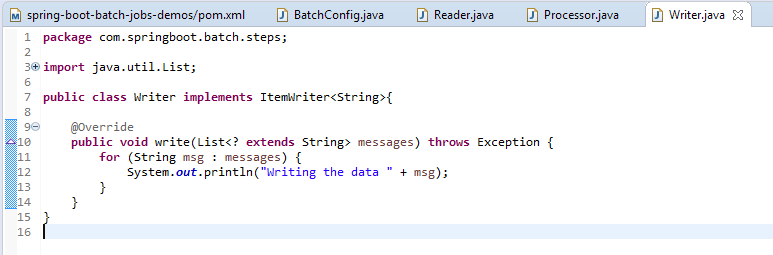
</build>

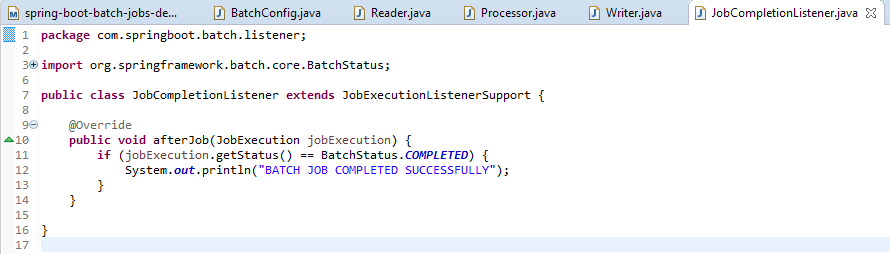
</project>



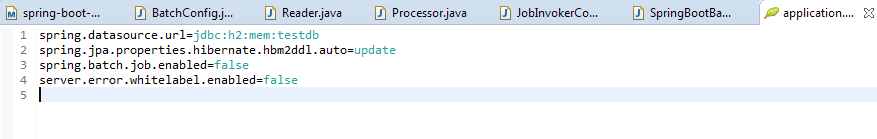


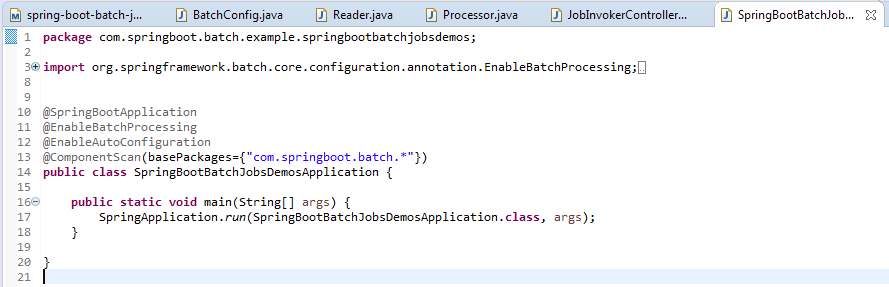


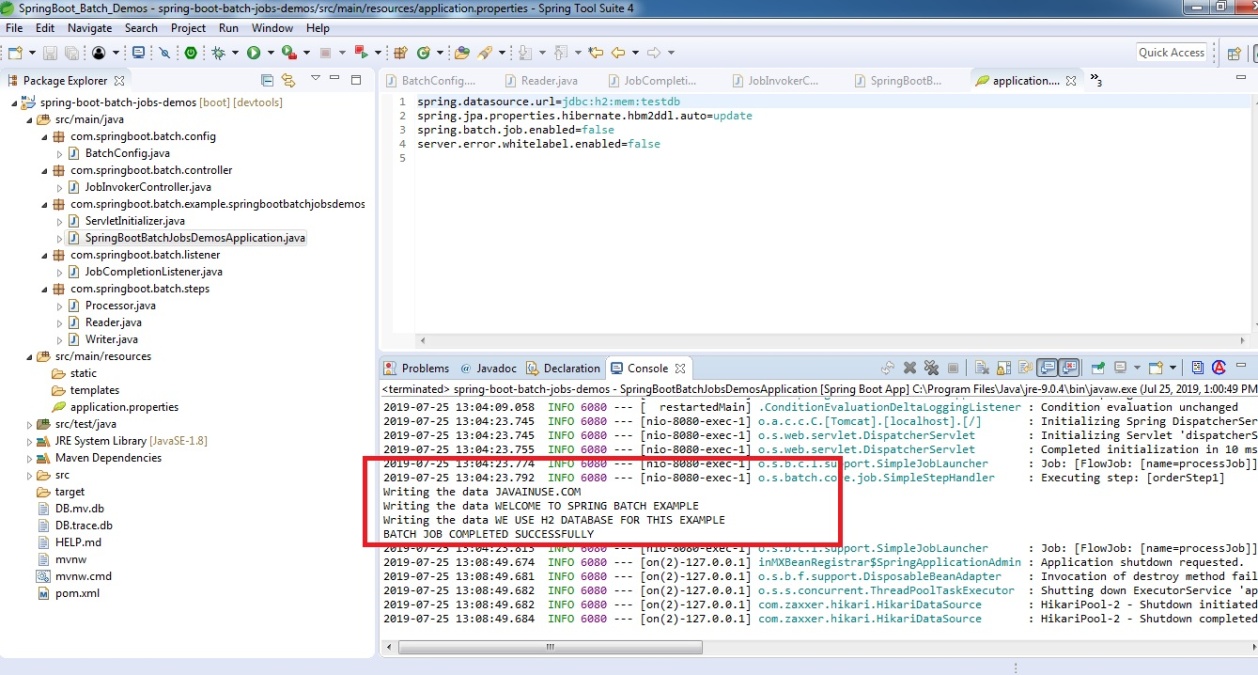


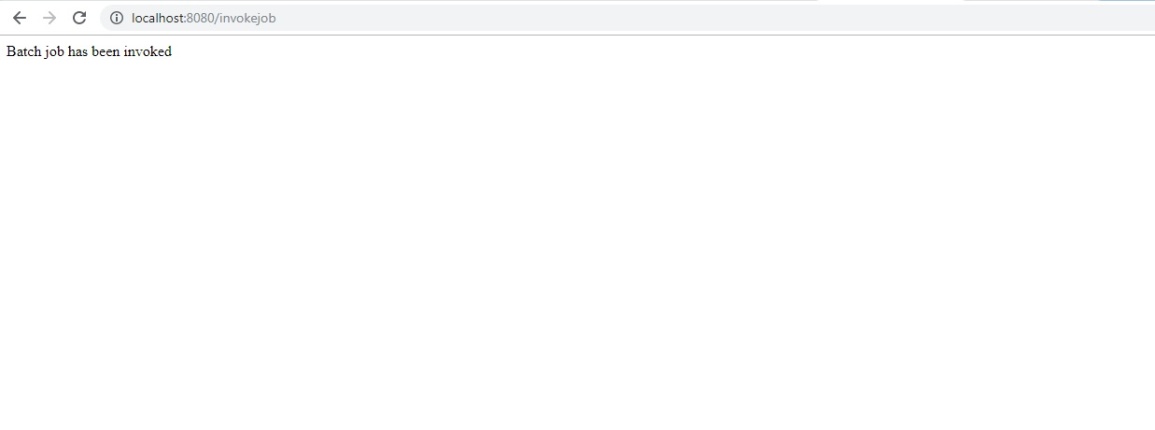


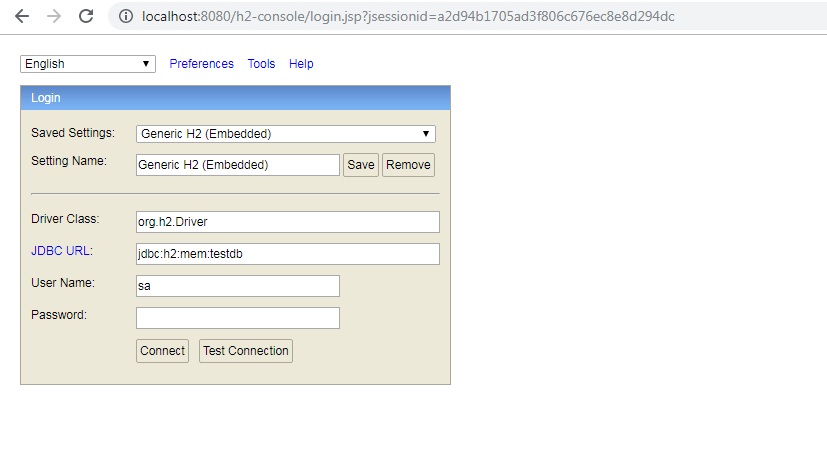


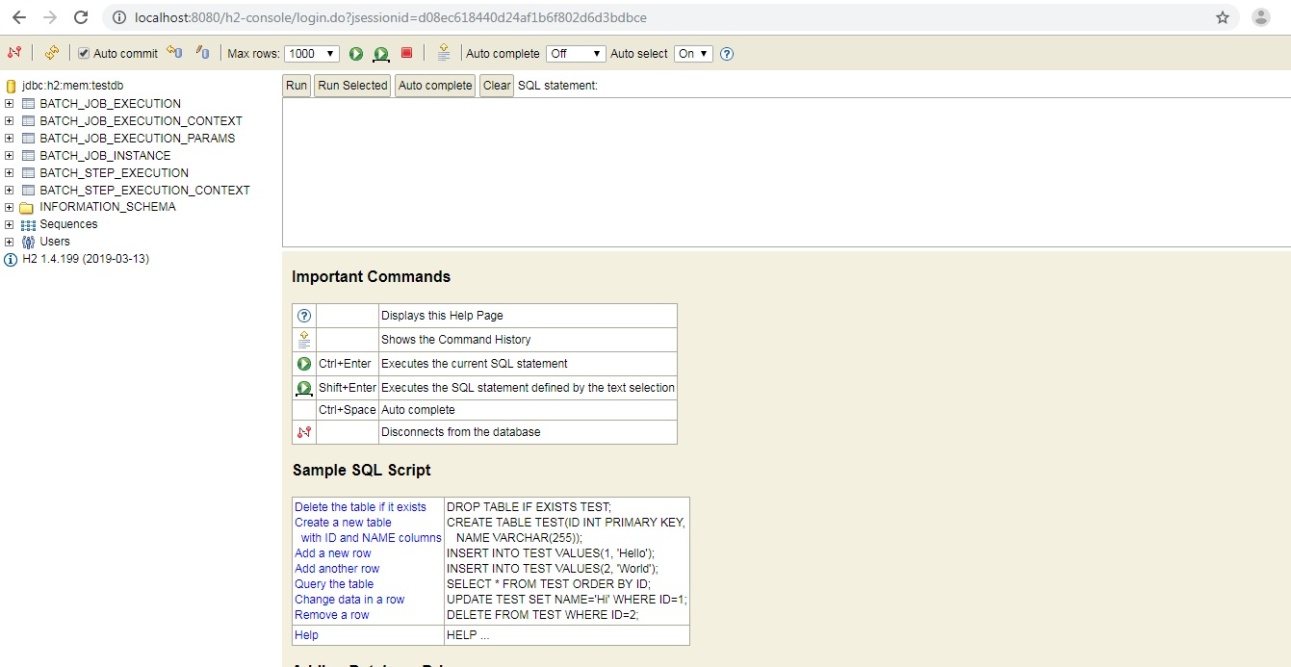


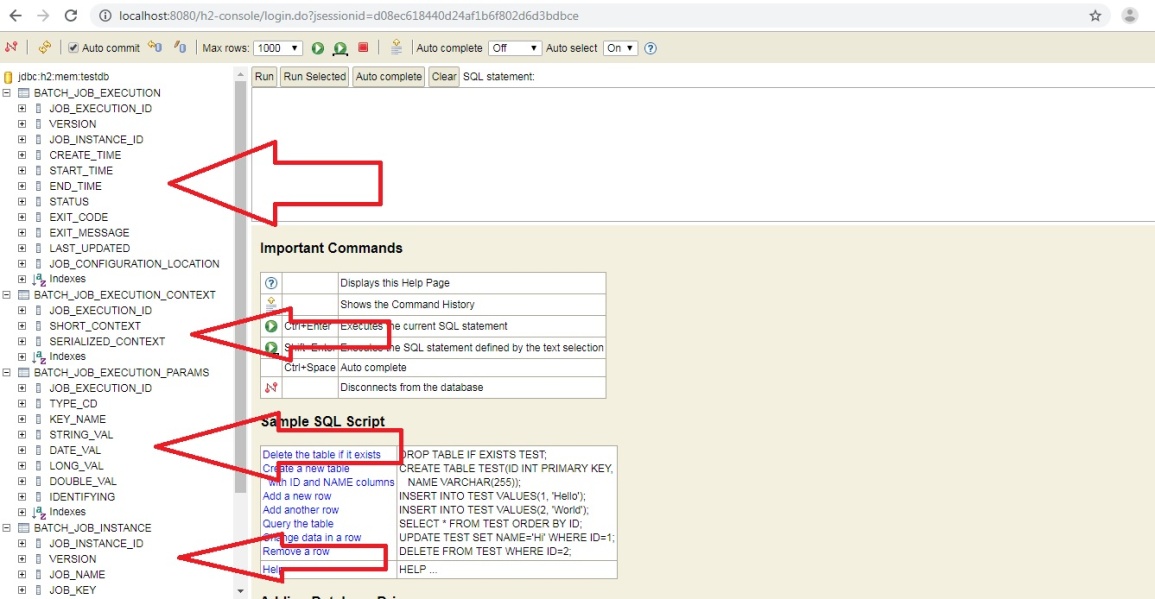












# Spring Batch - Difference between Step, Chunk and Tasklet:

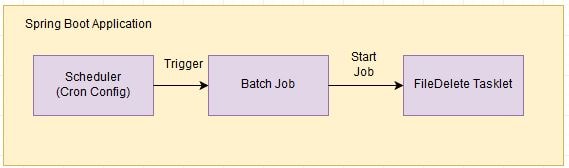
A **Step** is a domain object that encapsulates an independent, sequential phase of a batch job and contains all of the information necessary to define and control the actual batch processing.  
  
Steps can be processed in either of the following two ways.

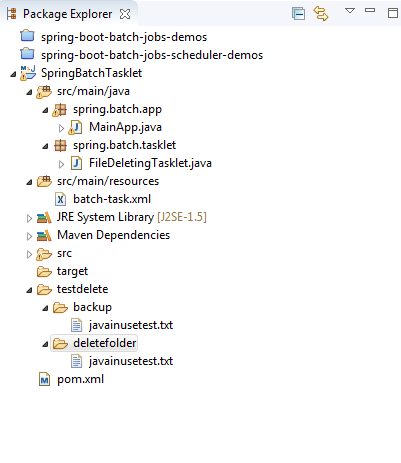
* Chunk
* Tasklet

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Tasklet** | **Chunk** |
| **When to use** | Suppose the job to be run a single granular task then Tasklet processing is used. | Suppose the job to be run is complex and involves executing of tasks involving reads, processing and writes the we use chunk oriented processing |
| **How it works** | No aggregation, just the task gets executed. | It involves reading an input, processing it based on the business logic and then aggregating it till the commit-interval is reached and finally writing out the chunk of data output to a file or database table. |
| **Usage** | It’s not used commonly. | It’s the most common way of executing a Step. |
| **Use Case** | Usually Used in scenarios involving a single task like deleting a resource or executing a query . | Usually used in scenarios where multiple aggregated steps need to be run like copying, processing and transferring of data . |
| **Example** | <job id="taskletJob"> <step id="callingStoredProc"> <tasklet ref="callProc"/> </step> </job> | <job id="sampleJob" job-repository="jobRepository"> <step id="step1"> <tasklet transaction-manager="transactionManager"> <chunk reader="itemReader" writer="itemWriter" commit-interval="10"/> </tasklet> </step> </job> |

# Spring Boot Batch Job + Scheduler Simple Example

In this example we develop a simple Spring Boot Batch application where batch job gets triggered using a scheduler.  
Consider the simple use case where the user wants to delete files from a particular location everyday at a particular time. We will schedule this batch job using the scheduler.





**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/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.springbatch.tasklet</groupId>

<artifactId>SpringBatchTasklet</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>SpringBatchTasklet Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<spring.version>3.2.2.RELEASE</spring.version>

<spring.batch.version>2.2.0.RELEASE</spring.batch.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>3.2.2.RELEASE</version>

</dependency>

<!-- Spring Batch dependencies :: BEGINS -->

<dependency>

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

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

<version>2.2.2.RELEASE</version>

</dependency>

<dependency>

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

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

<version>2.2.2.RELEASE</version>

</dependency>

<!-- Spring Batch dependencies :: ENDS-->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

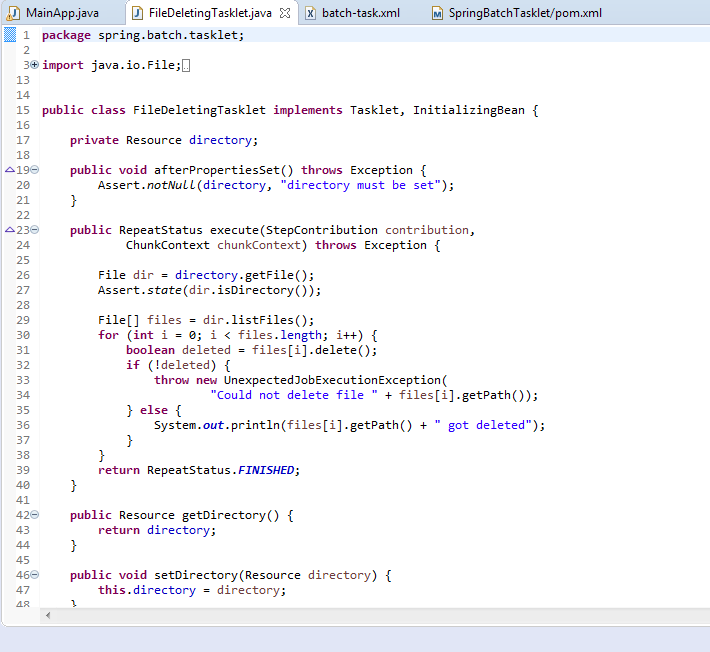
</dependencies>

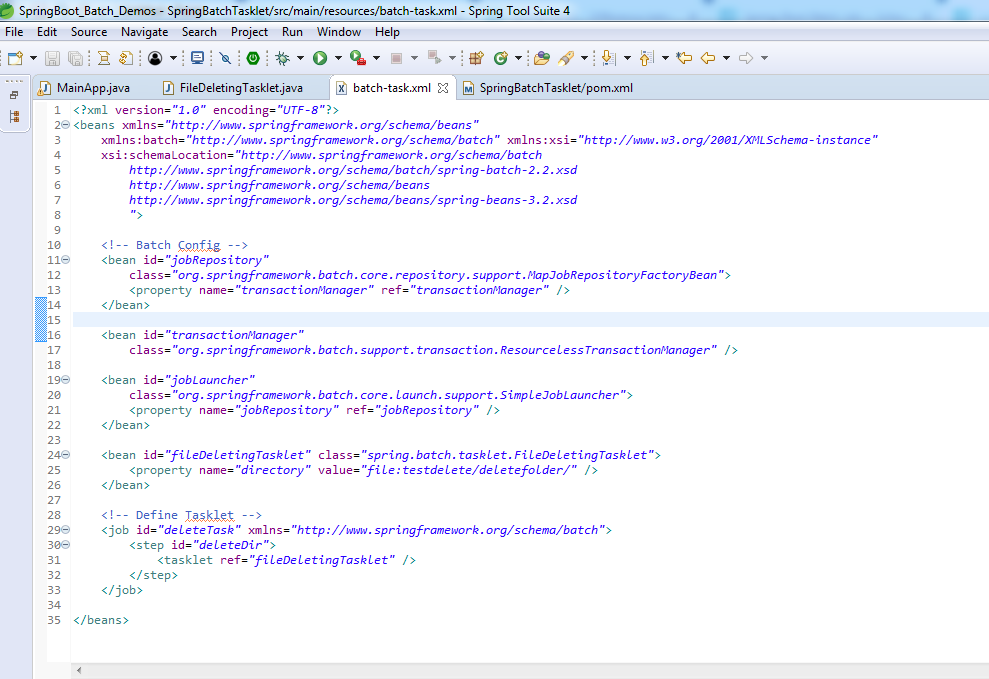
<build>

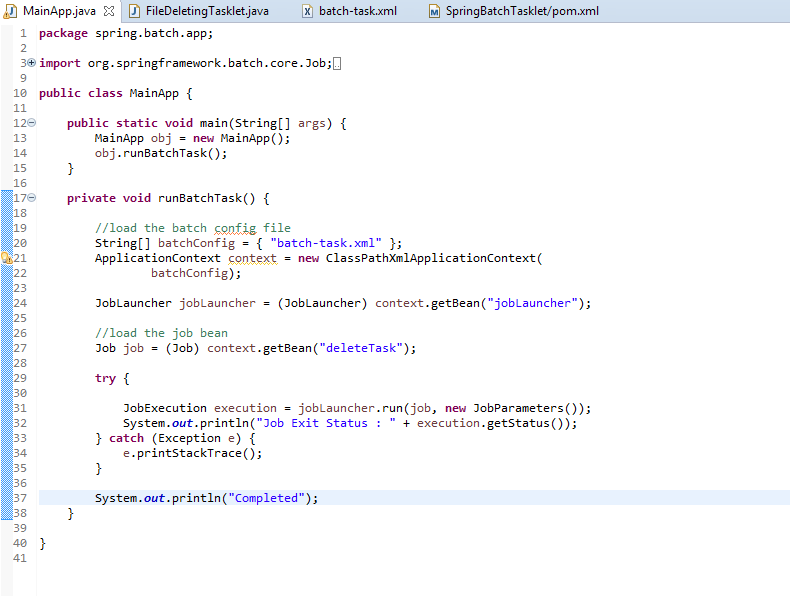
<finalName>SpringBatchTasklet</finalName>

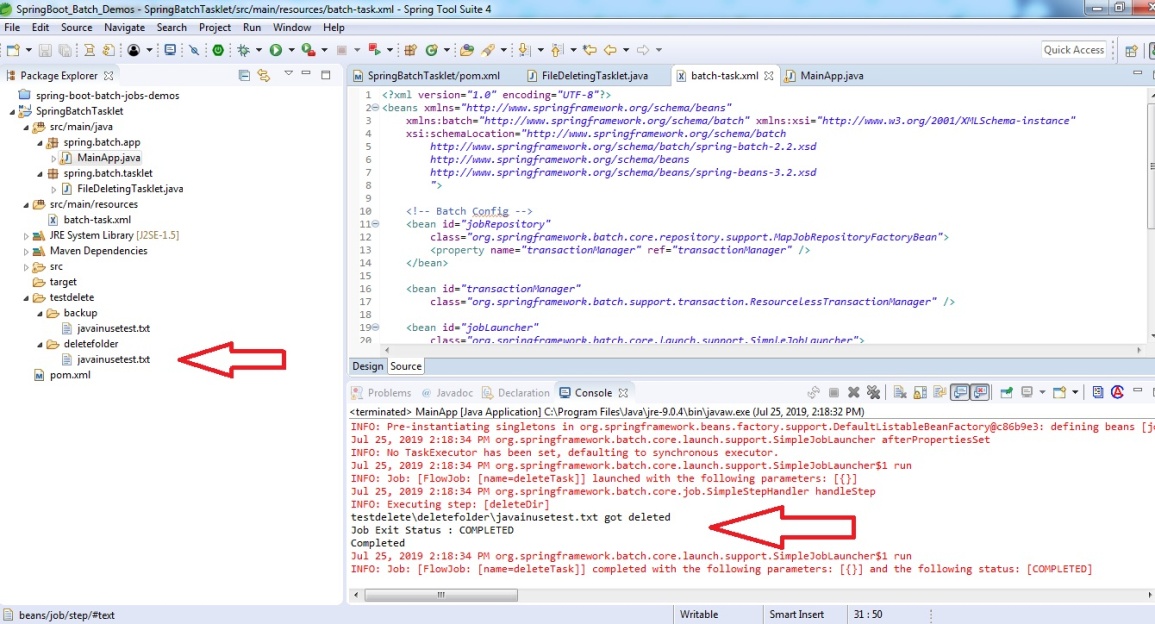
</build>

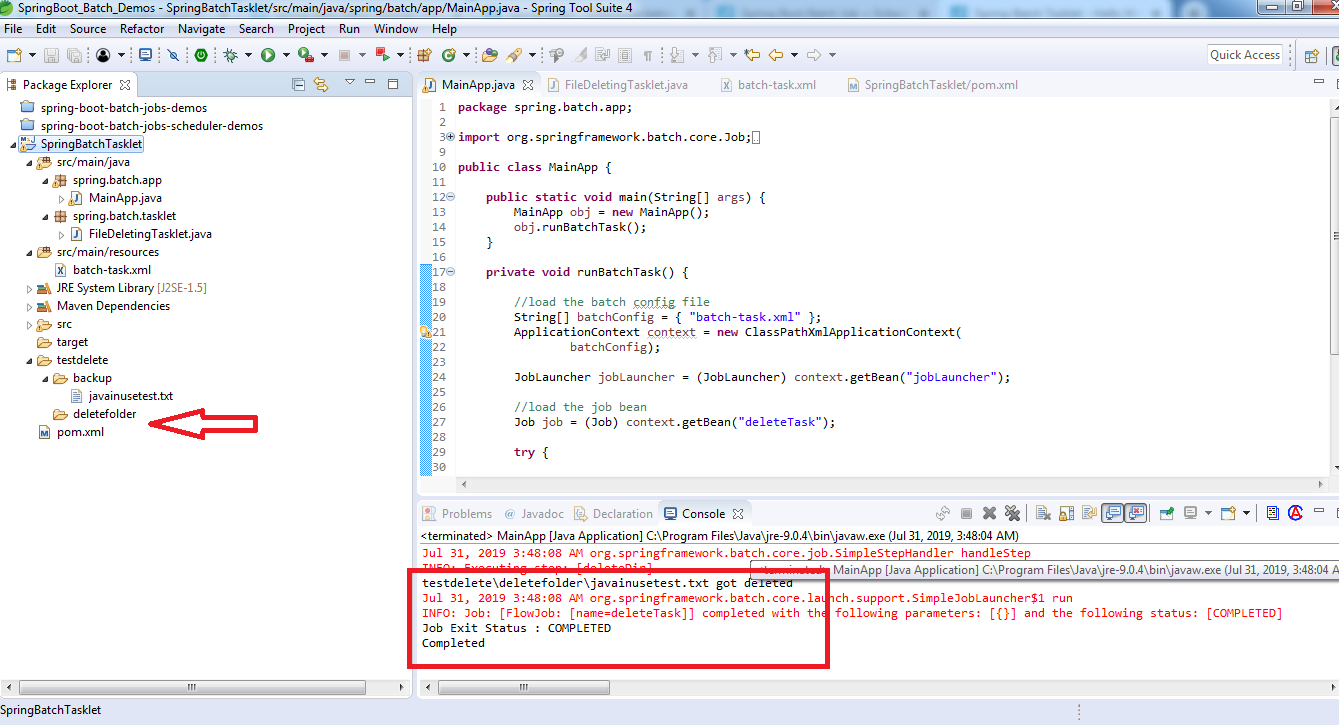
</project>



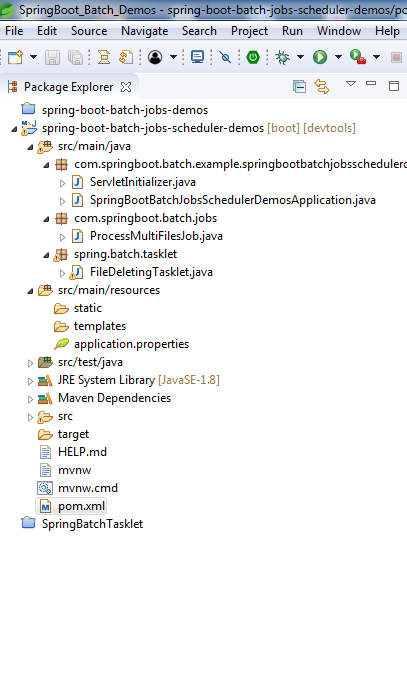








Example 2:



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

<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>

<parent>

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

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

<version>2.1.6.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.springboot.batch.example</groupId>

<artifactId>spring-boot-batch-jobs-scheduler-demos</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>spring-boot-batch-jobs-scheduler-demos</name>

<description>Demo project for Spring Boot</description>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>provided</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

