«

Spring / Spring batch Interview questions

»

Could not find what you were looking for? send us the question and we would be happy to answer your question.

Explain spring batch framework.

Spring Batch frame work, a collaborative effort from Accenture and SpringSource, is a lightweight, comprehensive framework that facilitates the development of batch applications that helps the day to day activities of enterprise systems. Batch application or processing refers to automated offline systems that performs bulk data processing, periodic updates and delegated processing.

Examples include loading csv file data to database, process feed file once received and push daily transactions to the upstream or downstream systems.

List out some of the practical usage scenario of Spring Batch framework.

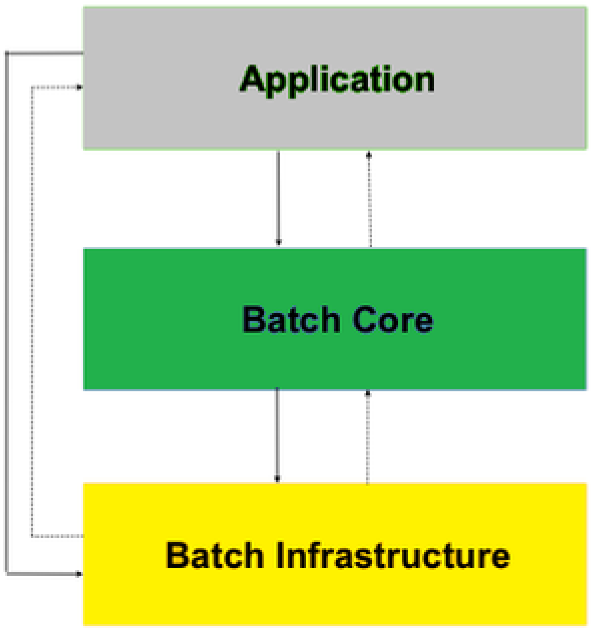
- Reading large number of records from a database, file, queue or any other medium, process it and store the processed records into medium, for example, database.
- Concurrent and massively parallel processing.
- Staged, enterprise message-driven processing.
- Sequential processing of dependent steps.
- Whole-batch transaction.
- Scheduled and repeated processing.

Technical advantages of using Spring Batch Framework from a Developer perspective.

- Batch framework leverages Spring programming model thus allows developers to concentrate on the business logic or the business procedure and framework facilitates the infrastructure.
- Clear separation of concerns between the infrastructure, the batch execution environment, the batch application and the different steps/proceses within a batch application.
- Provides common scenario based, core execution services as interfaces that the applications can implement and in addition to that framework provides its default implementation that the developers could use or partially override based on their business logic.
- Easily configurable and extendable services across different layers.
- Provides a simple deployment model built using Maven.

Explain the Spring Batch framework architecture.

Spring Batch exhibit a layered architecture and it comprises of three major high level components: Application, Core and Infrastructure.



The **application** layer contains all the batch job configurations, custom codes for business logic and job meta information developed by Application developers.

The Batch Core has the core runtime classes necessary to launch and control any batch job. Some of the core runtime classes include JobLauncher, Job, and Step implementations.

The **infrastructure** contains API for common readers and writers, and services for retrying on failure, repeat jobs etc. The infrastructure layer are used both by application developers(ItemReader and ItemWriter) and the core framework itself for controlling the batch job such as Retry, repeat. Thus Batch Core and Application layers are built on top of Infrastructure layer.

How do you categorize batch applications based on input source in spring batch?

- **Database-driven** applications are driven by rows or values received from the database.
- **File-driven** applications are driven by records or values retrieved from a file.

- **Message-driven** applications are driven by messages retrieved from a message queue.

What are the typical processing strategies in spring batch?

- Normal processing during offline.
- Concurrent batch or online processing.
- Parallel processing of many different batch or jobs at the same time.
- Partitioning (processing of many instances of the same job at the same time).
- A combination of the above.

How do I start a Spring Batch Job?

A Job Launcher can be used to execute a Spring Batch Job. Also a batch job can be launched/scheduled using a web container as well.

Execution of a job is termed as Job Instance. Each Job Instance is provided with an execution id which can be used to restart the job if required.

Job can be configured with parameters which is passed to it from the Job Launcher.

What are the important features of Spring Batch?

- **Restorability:** Restart a batch program from where it failed.
- **Different Readers and Writers :** Provides great support to read from text files, csv, JMS, JDBC, Hibernate, iBatis etc. It can write to JMS, JDBC, Hibernate, files and many more.
- **Chunk Processing :** If we have 1 Million records to process, these can be processed in configurable chunks (1000 at a time or 10000 at a time).
- Easy to implement proper transaction management even when using chunk processing.
- Easy to implement parallel processing. With simple configuration, different steps can be run in parallel.

Explain Normal processing strategy in spring Batch framework.

Normal processing refers to the batch processes that runs in a separate batch window, the data being updated is not required by on-line users or other batch processes, where concurrency would not be a concern and a single commit can be done at the end of the batch run.

Single commit point may be a concern in terms of scaiability and volume of data it could handle, it is always a good practice to have restart recovery options .

Explain Concurrent batch on-line processing in Spring Batch Framework.

Concurrent/on-line batch processing refers to the batch process that handles data being concurrently used/updated by online users so the data cannot be locked in database or file as the online users will need it. Also the data updates should be commited frequently at the end of few transactions to minimize the portion of data that is unavailable to other processes and the elapsed time the data is unavailable.

Explain parallel processing in Spring Batch Framework.

Parallel processing enables multiple batch runs jobs to run in parallel to reduce the total elapsed batch processing time. Parallel processing is simpler as long as the same file or database table is not shared among the processes otherwise the processes should process partitioned data.

Another approach would be using a control table for maintaining interdependencies and to track each shared resource in use by any process or not.

Other key issues in parallel processing include load balancing and the availability of general system resources such as files, database buffer pools etc. Also note that the control table itself can easily become a critical resource.

Explain Partitioning in Spring Batch framework.

Partitioning faciliates multiple large batch applications to run concurrently that minimize the elapsed time required to process long batch jobs. Processes which can be successfully partitioned are those where the input file can be split and/or the main database tables partitioned to allow the application to run against different sets of data.

Processes which are partitioned must be designed to only process their assigned data set.

Explain the spring batch framework architecture.

Job is an entity that encapsulates the entire batch process and acts as a container for steps. A job combines multiple steps that belong logically together in a flow and allows for configuration of properties global to all steps such as restartability.

A job is wired using an XML configuration or java based configuration and it is referred as Job configuration.

The job configuration contains:

- The name of the job.
- Definition and ordering of steps.
- Whether or not the job is restartable.

JobInstance represents a logical job run. Consider a batch job that runs every night, there is a logical JobInstance running everyday.

JobParameters are set of parameters used to start a batch Job. They also distinguish JobInstance from another.

What is tasklet in Spring batch framework?

The Tasklet is an interface which performs any single task such as setup resource, running a sql update, cleaning up resources etc.

How do I configure a job in spring batch framework?

A Job in Spring Batch contains a sequence of one or more Steps. Each Step can be configured with the list of parameters/attribute required to execute each step.

next : next step to execute

tasklet: task or chunk to execute. A chunk can be configured with a Item Reader, Item Processor and Item Writer.

decision : Decide which steps need to executed.

What are Spring Batch metadata schema?

The Spring Batch Meta-Data tables are used to persist batch domain objects such as JobInstance, JobExecution, JobParameters, and StepExecution for internally managing the Batch Jobs.

The JobRepository is responsible for saving and storing each Java object into its correct table.

Can we create a Spring batch with No step?

No. There must exists at least one step or flow or split configuration within a Spring Batch job.

what are the different bean scope in Spring Batch 3.0?

Step scope- there is only one instance of such a bean per executing step.

```
<bean id="..." class="..." scope="step">
```

Job scope- there is only one instance of such a bean per executing Job.

```
<bean id="..." class="..." scope="job">
```

Can we configure Spring Batch without persisting metadata to database?

Yes, it is configurable. We need to use **MapJobRepositoryFactoryBean** and **ResourcelessTransactionManager**.

```
<bean id="appTransactionManager"
    class="org.springframework.batch.support.transaction.ResourcelessTransactionManager" />

<bean id="appJobRepository"
    class="org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean">
    <property name="transactionManager" ref="transactionManager" />
</bean>

<bean id="appJobLauncher"
    class="org.springframework.batch.core.launch.support.SimpleJobLauncher">
    <property name="jobRepository" ref="appJobRepository" />
</bean>
```

How do we track the number of item processed by the ItemReader in Spring batch?

The item mapping bean can implement org.springframework.batch.item. **ItemCountAware**, a marker interface to have the item position tracked.

Define a Job in Spring Batch.

A Job is an entity that encapsulates an entire batch process.

Job will be wired together using a XML configuration file or Java based configuration. This configuration is also referred as "job configuration".

A Job is simply a container for Steps and it combines multiple steps that runs logically together in a flow.

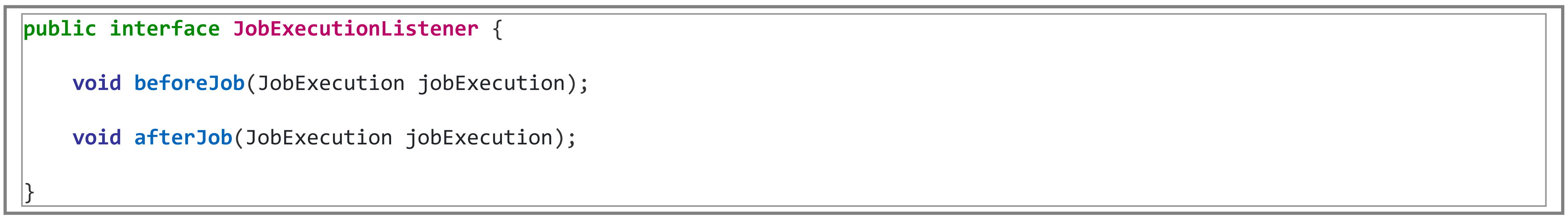
What are the elements of job configuration in spring batch?

The job configuration allows for configuration of properties global to all steps. The job configuration contains,

- name of the job.
- Definition and ordering of Steps .
- Whether or not the job is restartable.

How do I intercept a Job Execution in Spring Batch?

During the course of the execution of a Job, it may be useful to be notified of various events in its lifecycle so that custom code may be executed. The SimpleJob allows for this by calling a JobListener at the appropriate time.



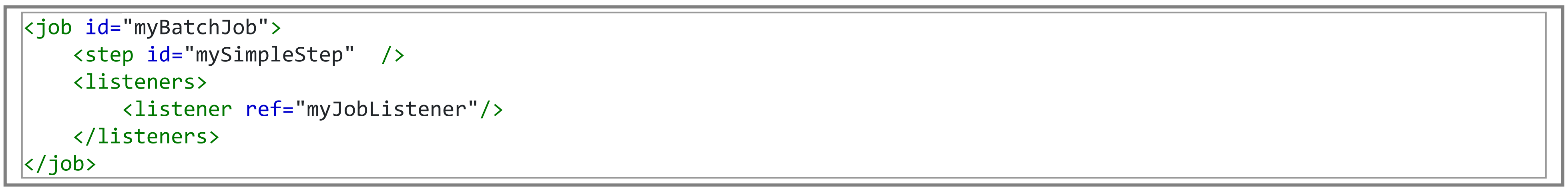
```
public interface JobExecutionListener {

    void beforeJob(JobExecution jobExecution);

    void afterJob(JobExecution jobExecution);

}
```

JobListeners can be added to a SimpleJob via the listeners element on the job.



```
<job id="myBatchJob">
    <step id="mySimpleStep"  />
    <listeners>
        <listener ref="myJobListener"/>
    </listeners>
</job>
```

The annotations corresponding to this interface are:

- @BeforeJob.
- @AfterJob.

What is Job launcher in Spring Batch framework?

JobLauncher represents a simple interface for launching a Job with a given set of JobParameters.

Spring Batch Framework: What is ExecutionContext?

An ExecutionContext represents a collection of key/value pairs that are persisted and controlled by the framework in order to provide the developers a placeholder to store persistent state that is scoped to a StepExecution or JobExecution.

Can we create Spring Batch Job without using XML configuration?

Spring 3 brought the ability to configure applications via java instead of XML. As of Spring Batch 2.2.0, batch jobs can be configured using the same java config. There are two components for the java based configuration: the **@EnableBatchConfiguration** annotation and two builders, **JobBuilderFactory** and the **StepBuilderFactory**.

```
@Configuration
@EnableBatchProcessing
@Import(DataSourceConfiguration.class)
public class SimpleJobConfig {
    private static final String JOB_NAME = "DynamicJob";

    @Autowired
    private JobBuilderFactory jobs;

    @Autowired
    private StepBuilderFactory steps;


    @Bean
    public Job job() throws Exception {
        SimpleJobBuilder jobBuilder = this.jobs.get(createJobName(JOB_NAME)).start(firstStep());

        for ... {
            jobBuilder.next(createAStep( ... params ...));
        }

        return standardJob.build();
    }

    public Step createAStep(String name, ... needed parameters ...) throws Exception {
        StepBuilder stepBuilder = this.steps.get(name);

        return standardStep1.build();
    }
}
```

How do you run Spring Batch jobs in Production Environment?

Usually The Java batch Job main class and its dependencies are passed to the java command and it is stored in a command line Batch file or shell script in terms of linux/unix.

These script file can be run using scheduler like Autosys at the Production environment.

What is CommandLineJobRunner in Spring Batch?

CommandLineJobRunner is one of the ways to bootstrap your Spring batch Job. The xml script launching the job needs a Java class main method as as entry point and CommandLineJobRunner helps you to start your job directly using the XML script.

The CommandLineJobRunner performs 4 tasks.

- Load the appropriate ApplicationContext.
- Parse command line arguments into JobParameters.
- Locate the appropriate job based on arguments.
- Use the JobLauncher provided in the application context to launch the job..

The CommandLineJobRunner arguments are **jobPath**, the location of the XML file that will be used to create an ApplicationContext and the **jobName**, the name of the job to be run.

```
bash$ java CommandLineJobRunner DailyJobConfig.xml processDailyJob
```

These arguments must be passed in with the path first and the name second. All arguments after these are considered to be JobParameters and must be in the format of **'name=value'**.

How do I configure parallel execution of steps in Spring Batch?

We may use split tag to configure parallel steps. In below example, flow1 (step1 and step2 sequence) is executed in parallel with flow2 (step3). Step4 is executed after both flows are complete.

```
<job id="parallelJob">
    <split id="split1" task-executor="taskExecutor" next="step4">
        <flow>
            <step id="step1" parent="s1" next="step2"/>
            <step id="step2" parent="s2"/>
        </flow>
        <flow>
            <step id="step3" parent="s3"/>
        </flow>
    </split>
    <step id="step4" parent="s4"/>
</job>
```

What is ResourceAware is Spring Batch?

ResourceAware is a marker interface which will set the current resource on any item that implement this interface.

Difference between Spring Batch and Quartz Scheduler.

Spring Batch and Quartz have different features and responsibility. Spring Batch provides functionality for processing large volumes of data while Quartz provides functionality for scheduling tasks. Thus Quartz could complement Spring Batch and a common combination would be to use Quartz as a trigger for a Spring Batch job using a Cron expression.

How do I schedule a job with Spring Batch?

Use a scheduling tool such as Quartz, Control-M or Autosys. Quartz islight weight, doesn't have all the features of Control-M or Autosys. Even the OS based Task scheduler, CRON jobs could be used to schedule Spring batch jobs.

How can I make an item reader thread safe in Spring Batch?

You can **synchronize the read()** method. Remember that you will lose restartability, so best practice is to mark the **step as not restartable** and to be safe (and efficient) you can also set **saveState=false** on the reader.

What is the latest version of Spring Batch?

The available latest version is 4.0.1.

What is ItemReader in Spring batch framework?

ItemReader is an abstraction that represents the retrieval of input for a Step, one item/row/record at a time. When the ItemReader has exhausted the items it can provide, it will indicate this by returning null.

What is ItemWriter in Spring batch framework?

ItemWriter is an abstraction that represents the output of a Step, one batch or chunk of items at a time. Generally, an item writer has no knowledge of the input it will receive next, only the item that was passed in its current invocation.

Spring Batch: What is ItemProcessor?

ItemProcessor is an abstraction that represents the business processing of an item. While the ItemReader reads one item, and the ItemWriter writes them, the ItemProcessor provides access to transform or apply other business processing. If, while processing the item, it is determined that the item is not valid, returning null indicates that the item should not be written out.

Mention the different ItemReader and ItemWriter implementations available in Spring Batch?

There are many implementations including the ones that allow read and write operations on,

- Flat File.
- Xml.
- Hibernate Cursor.
- JDBC.
- JMS.
- Hibernate Paging.
- Stored Procedure.

Name few of the domain buzzwords in Spring Batch.

- Job.
- JobLauncher.
- JobRepository.
- JobInstance.
- JobExecution.
- JobParameters.

Spring Batch: What Job configuration consists of?

The job configuration contains,

- The simple name of the job.
- Definition and order of the Steps.
- configurable global(to all steps) properties such as restartablity.

Spring Batch: What is JobInstance?

A JobInstance represents the concept of a logical job run.

Spring Batch: Difference between Step and StepExecution.

A Step is a domain object that encapsulates an independent and sequential phase of a batch job while a StepExecution represents a single attempt to execute a step.

Spring batch: Define ExecutionContext.

An ExecutionContext represents a collection of key-value pairs that are persisted and controlled by the framework in order to allow developers a place to store persistent state that is scoped to a StepExecution or JobExecution.

Spring batch: uses of JobRepository.

JobRepository is used for basic CRUD operations of the various persisted domain objects within Spring Batch, such as JobExecution and StepExecution. It is required by many of the major framework features, such as the JobLauncher, Job, and Step.

```
<job-repository id="myJobRepository"
    data-source="myDataSource"
    transaction-manager="myTransactionManager"
    isolation-level-for-create="SERIALIZABLE"
    table-prefix="MyBATCH_"
        max-varchar-length="1000"/>
```

The only required property is "id" and others are optional.

Spring batch: What are the required dependencies for configuring a job?

There are 3 required dependencies,

- Job name,
- JobRepository,
- and one or more steps.

Spring batch: list the job configurations other than step.

In addition to steps a job configuration can contain other elements such as,

- parallelisation (<split/>),
- declarative flow control (<decision/>)

- and externalization of flow definitions (<flow/>).

Can a spring batch Job have a parent job?

If a group of Jobs share similar, but not identical, configurations, then it may be helpful to define a "parent" Job from which the concrete Jobs may inherit properties.

```
<job id="parentJob" abstract="true">
    <listeners>
        <listener ref="parentListener"/>
    <listeners>
</job>

<job id="childJob" parent="parentJob">
    <step id="step1" parent="standaloneStep"/>

    <listeners merge="true">
        <listener ref="listenerTwo"/>
    <listeners>
</job>
```

Explain the role of JobParametersValidator in Spring Batch.

A job declared in the XML namespace or using any subclass of AbstractJob can optionally declare a validator for the job parameters at runtime that helps to assert that a job is started with all its mandatory parameters. There is a DefaultJobParametersValidator that can be used to constrain combinations of simple mandatory and optional parameters, and for more complex constraints you can implement the interface by yourself. The configuration of a validator is supported through the XML namespace through a child element of the job.

```
<job id="job1" parent="baseJob">
    <step id="step1" parent="baseStep"/>
    <validator ref="paremetersValidator"/>
</job>
```

How do I setup Spring batch job without using XML?

Spring 3 enables the ability to configure applications using java instead of XML and from Spring Batch 2.2.0, batch jobs can be configured using the same java config.

There are 2 components for the java based configuration: the @EnableBatchConfiguration annotation and two builders. @EnableBatchProcessing provides a base configuration for building batch jobs. The core interface for this configuration is the BatchConfigurer. The default implementation provides the beans to be autowired such as JobRepository, JobLauncher.

Explain the role of JobRepository in Spring batch.

The JobRepository is used for basic CRUD operations of the various persisted domain objects within Spring Batch, such as JobExecution and StepExecution. It is required by many of the major framework features, such as the JobLauncher, Job, and Step.

How do you implement conditional step flow in Spring batch?

Next element is an transitional element in step that instructs the Job which Step to execute next.

```
<job id="job">
    <step id="stepA" parent="s1">
        <next on="*" to="stepB" />
        <next on="FAILED" to="stepC" />
    </step>
    <step id="stepB" parent="s2" next="stepC" />
    <step id="stepC" parent="s3" />
</job>
```

The "on" attribute of a transition element uses a simple pattern-matching scheme to match the ExitStatus that results from the execution of the Step. Only two special characters are allowed in the pattern: "*" will zero or more characters and "?" will match exactly one character.

What is the default isolation level of Spring batch transactions?

It is **SERIALIZABLE** by default to prevent the same job instance being executed concurrently.

How do I configure parallel execution of steps in Spring batch?

The **split** configuration can be used to execute a job steps in parallel. In the below example flow1 and flow2 are executed parallely.

```
<job id="myJob">
    <split id="split1" task-executor="taskExecutor" next="step4">
        <flow id="flow1">
            <step id="step1" parent="s1" next="step2"/>
            <step id="step2" parent="s2"/>
        </flow>
        <flow id="flow2">
            <step id="step3" parent="s3"/>
        </flow>
    </split>
    <step id="step4" parent="s4"/>
</job>
```

What is a cron job?

A cron job is a Linux command for scheduling script on your server to execute repetitive tasks automatically. Scripts executed as a cron job are typically used to modify files, databases and manage caching.

How cron job works in Linux?

Cron is a daemon that executes scheduled commands. Cron is started automatically from /etc/init.d on entering multi-user runlevels. Cron searches its spool area (/var/spool/cron/crontabs) for crontab files (which are named after accounts in /etc/passwd); crontabs found are loaded into memory.

Cron wakes up every minute, examining all stored crontabs, checking each command to see if it should be run in the current minute. When executing commands, any output is mailed to the owner of the crontab (or to the user named in the MAILTO environment variable in the crontab, if such exists).

What is an autosys box?

A box is a container of jobs with similar starting conditions (date and time conditions or job dependency conditions). Use boxes to group jobs with similar scheduling parameters, not to group jobs organizationally.

Jobs in a box run only once for each box execution. Jobs in a box start only if the box itself has a status of RUNNING.

Jobs in a box start only if the box itself has a status of RUNNING. A box returns a status of SUCCESS when all the jobs it contains have run and returned a status of SUCCESS.

Custom Search

«

»

Comments & Discussions

0 Comments **www.javapedia.net** 1 **Login**

♡ **Recommend** **Share** **Sort by Best**

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON **WWW.JAVAPEDIA.NET**

Explain Spring Framework.

1 comment • a year ago

Javatutorials — 354 ts

How to prevent a child class from being serialized when it's parent class already implements …

2 comments • 6 months ago

Avatar kavya tg — Use of marker interface in javaMarker Interfaces are used to indicate something to compiler/JVM. If JVM see that a class is a object of …

Spring Boot interview questions

1 comment • a year ago

Javatutorials — Thank you Mayur Kohli for your comment and much appreciated.