Spring Batch Introduction

14th Feb 2013

Petr Zapletal
Common Application Development

Agenda

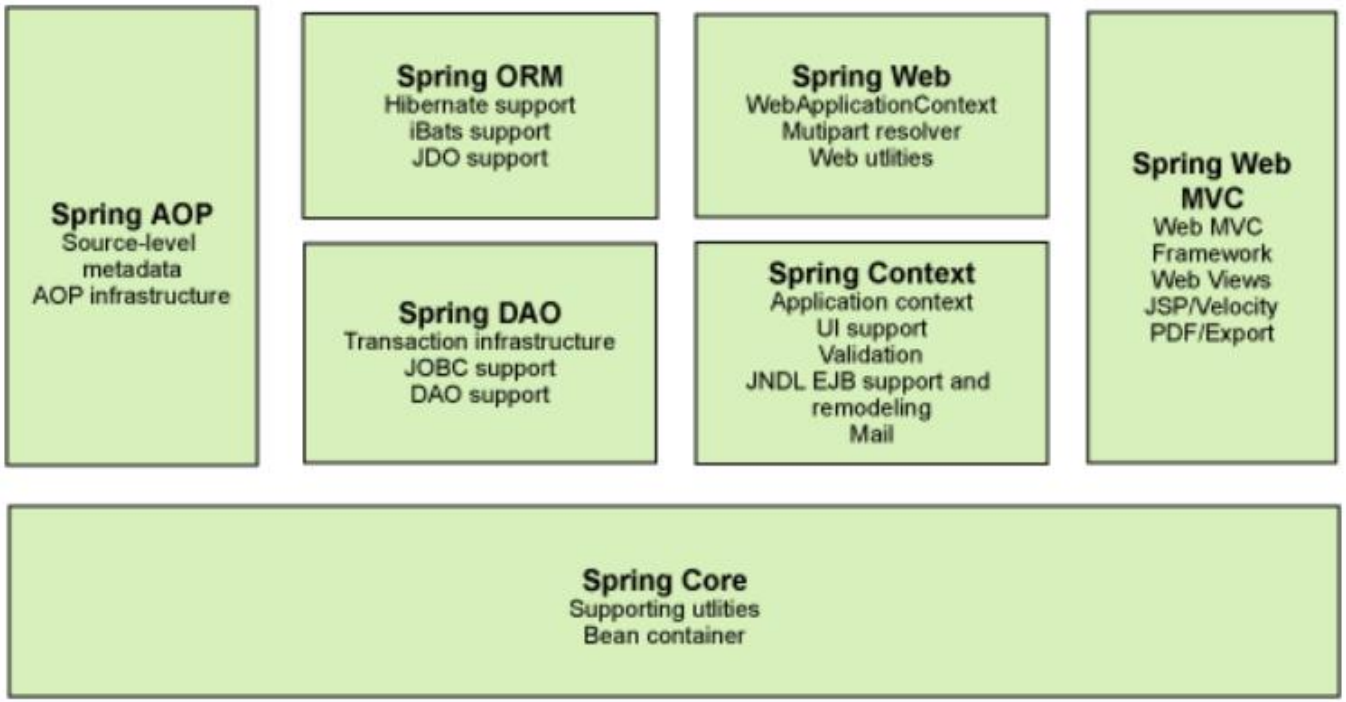
clearstream | DEUTSCHE BÖRSE GROUP

- Spring & Spring Batch Overview
- Domain Language of Batch
- Configuring and Running a Job
- Advanced Features
- Summary

Spring Overview

clearstream | DEUTSCHE BÖRSE GROUP

- One of the most popular open-source application development framework for enterprise Java
- Provides lightweight container



Spring Batch Introduction

clearstream | DEUTSCHE BÖRSE GROUP

- Java based framework for batch processing
- A lightweight, comprehensive batch framework
- Builds upon the productivity, POJO-based development approach, known from the Spring Framework
- Current version is 2.1.9.RELEASE

Batch Processing

clearstream | DEUTSCHE BÖRSE GROUP

- What is a Batch Application?
  - Batch applications need to process high volume business critical transactional data
  - A typical batch program generally
    1. reads a large number of records from a database, file, or queue
    2. processes the data in some fashion, and
    3. then writes back data in a modified form

Usage Scenarios

clearstream | DEUTSCHE BÖRSE GROUP

- Commit batch process periodically
- Concurrent batch processing: parallel processing of a job
- Staged, enterprise message-driven processing
- Massively parallel batch processing
- Manual or scheduled restart after failure
- Sequential processing of dependent steps
- Partial processing: skip records
- Whole-batch transaction: for cases with a small batch size or existing stored procedures/scripts

Agenda

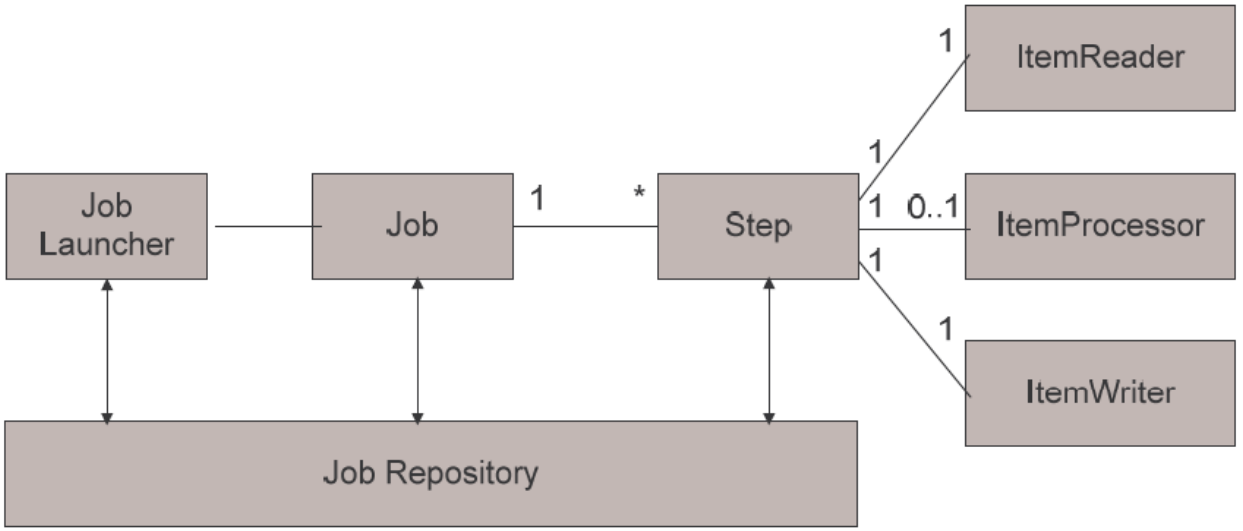
clearstream | DEUTSCHE BÖRSE GROUP

- Spring & Spring Batch Overview
- Domain Language of Batch
- Configuring and Running a Job
- Advanced Features
- Summary

Domain Language of Batch I

clearstream | DEUTSCHE BÖRSE GROUP

- A job has one to many steps
- A step has exactly one ItemReader, ItemWriter and optionally and ItemProcessor
- A job needs to be launched (JobLauncher)
- Meta data about the running process needs to be stored (JobRepository)



Domain Language of Batch II

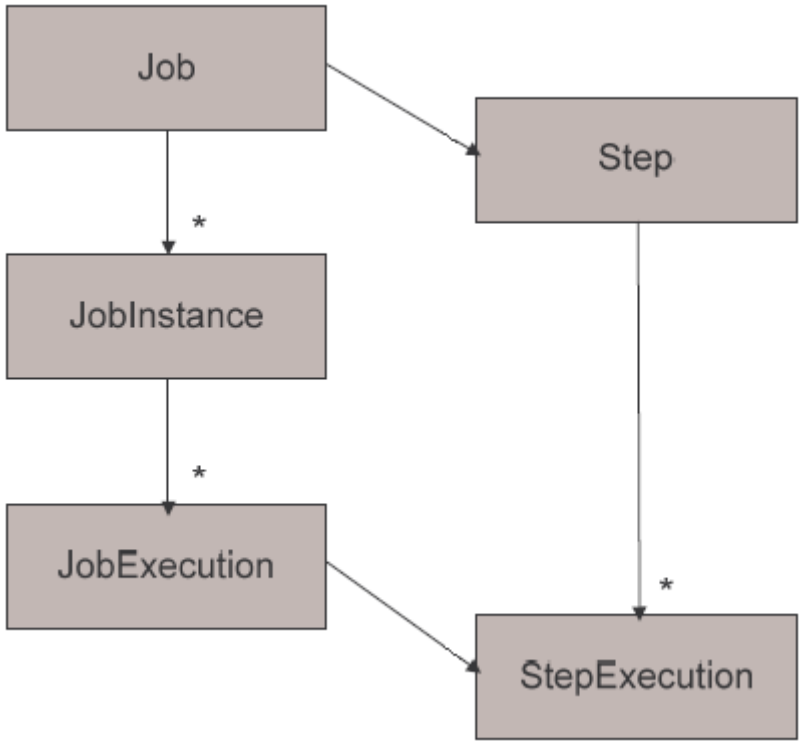
clearstream | DEUTSCHE BÖRSE GROUP

- Job encapsulates an entire batch process
- Job Instance
  - Refers to the concept of a logical job run
  - Job running once at end of day, will have one logical JobInstance
  - Each JobInstance can have multiple executions
- Job Execution
  - Refers to the technical concept of a single attempt to run a Job
  - An execution may end in failure or success, but the JobInstance will not be considered complete unless the execution completes successfully
- Job Parameters
  - Is a set of parameters used to start a batch job
  - JobInstance = Job + JobParameters

Domain Language of Batch III

clearstream | DEUTSCHE BÖRSE GROUP

- Step
  - A domain object that encapsulates an independent, sequential phase of a batch job
  - Can be as simple or complex as the developer desires
- Step Execution
  - Represents a single attempt to execute a Step
  - A new StepExecution will be created each time a Step is run, similar to JobExecution
  - A StepExecution will only be created when its Step is actually started



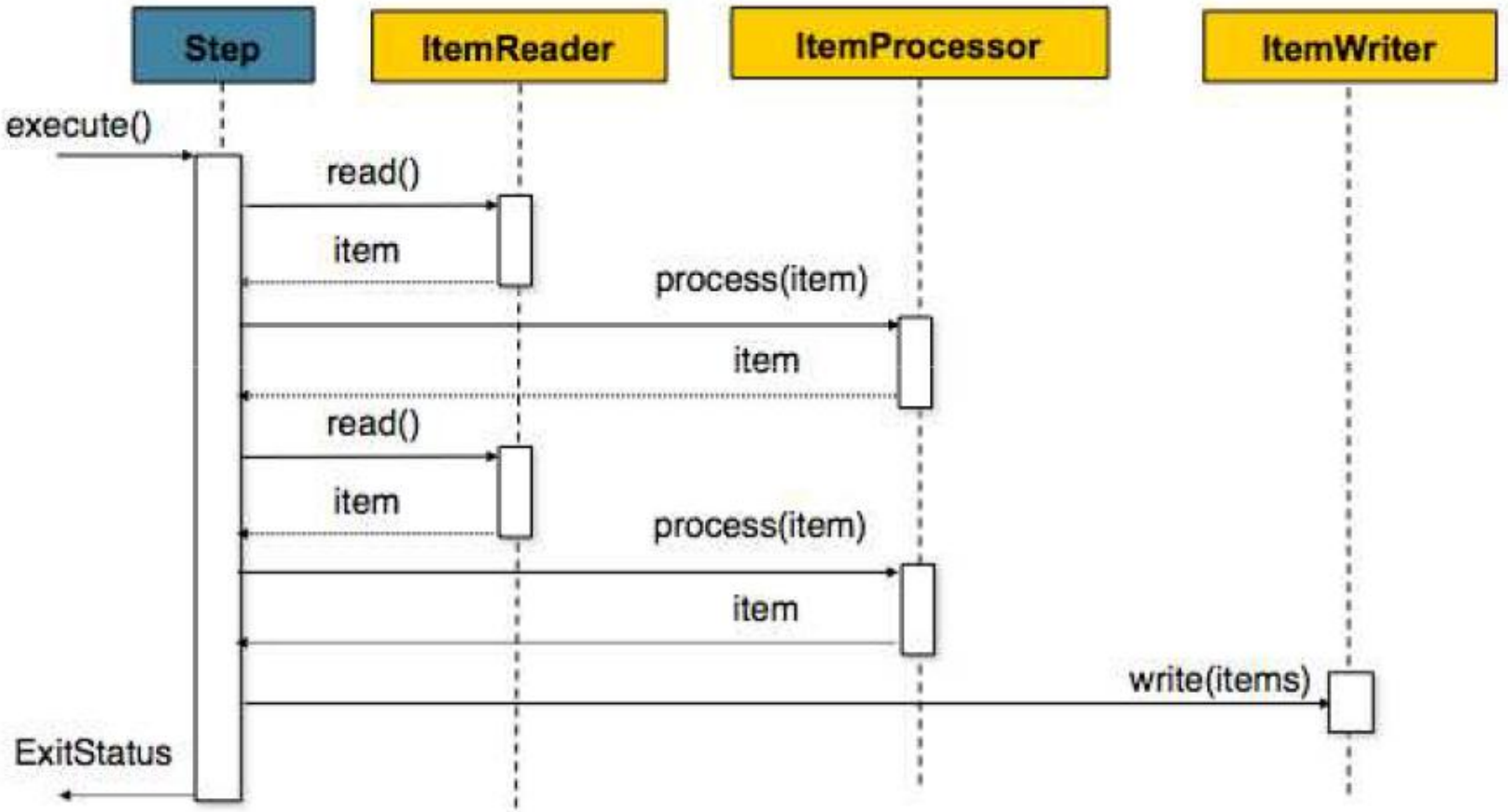
Domain Language of Batch IV

clearstream | DEUTSCHE BÖRSE GROUP

- Item Reader
  - An abstraction that represents the retrieval of input for a Step
  - When it has exhausted the items it can provide, it will indicate this by returning null
  - Various implementation available
- Item Writer
  - An abstraction that represents the output of a Step
  - Generally, an item writer has no knowledge of the input it will receive next
  - Various implementation available
- Item Processor
  - An abstraction that represents the business processing of an item
  - Provides access to transform or apply other business processing

Readers & Writers & Processors

clearstream | DEUTSCHE BÖRSE GROUP



Readers & Writers & Processors

clearstream | DEUTSCHE BÖRSE GROUP

– ItemReader

```
public interface ItemReader<T> {
    T read() throws Exception, UnexpectedInputException,
                    ParseException;
}
```

– ItemWriter

```
public interface ItemWriter<T> {
    void write(List<? extends T> items) throws Exception;
}
```

– ItemProcessor

```
public interface ItemProcessor<I, O> {
    O process(I item) throws Exception;
}
```

Job Repository & Job Launcher

clearstream | DEUTSCHE BÖRSE GROUP

- Job Repository
  - Batch persistence mechanism
  - Database and In-Memory implementation
  - Data about job executions, start times, durations, results, …
- Job Launcher
  - Represents a simple interface for launching a Job with a given set of JobParameters

Agenda

clearstream | DEUTSCHE BÖRSE GROUP

- Spring & Spring Batch Overview
- Domain Language of Batch
- Configuring and Running a Job
- Advanced Features
- Summary

Job Definition

clearstream | DEUTSCHE BÖRSE GROUP

– Configuring a Job and its steps

```
<job id="sampleJob">
  <step id="step1" job-repository="jobRepository"
        transaction-manager="transactionManager">
    <tasklet reader="itemReader" writer="itemWriter"
             commit-interval="10"/>
  </step>
</job>

<bean id="itemReader" ...>
<bean id="itemWriter" ...>
```

Job Repository & Job Launcher Definition

clearstream | DEUTSCHE BÖRSE GROUP

– Configuring a Job Repository

```
<job-repository id="jobRepository"
                dataSource="dataSource"
                transactionManager="transactionManager"
                isolation-level-for-create="serializable"
                table-prefix="BATCH_" />
```

– Configuring a Job Launcher

```
<bean id="jobLauncher"
    class="...batch.execution.launch.SimpleJobLauncher">
  <property name="jobRepository" ref="jobRepository" />
</bean>
```

Demo

```
<bean id="propertyPlaceholderConfigurer"
    class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE" />
    <property name="locations">
        <list>
            <value>classpath:batch.properties</value>
        </list>
    </property>
</bean>

<!-- NCCIP LOADER JOB -->
<job id="load-nccip" xmlns="http://www.springframework.org/schema/batch"
    job-repository="jobRepository-memory">
    <step id="nccipStep">
        <tasklet task-executor="taskExecutor" transaction-manager="transactionManager">
            <chunk reader="nccip-reader" writer="intact-jpa-writer"
                commit-interval="${batch.config.commit-interval}"
                retry-limit="1" skip-policy="skipPolicy">
                <retryable-exception-classes>
                    <include class="org.springframework.dao.TransientDataAccessException" />
                </retryable-exception-classes>
                <listeners>
                    <listener ref="skipListener" />
                </listeners>
            </chunk>
        </tasklet>
    </step>
    <listeners>
        <listener ref="intactJobExecutionListener" />
    </listeners>
</job>

<!-- RTS LOADER JOB -->
<job id="load-rts" xmlns="http://www.springframework.org/schema/batch"
    job-repository="jobRepository-memory">
    <step id="rtsStep">
        <tasklet task-executor="taskExecutor" transaction-manager="transactionManager">
            <chunk reader="rts-reader" writer="intact-jpa-writer"
                commit-interval="${batch.config.commit-interval}"
                retry-limit="1" skip-policy="skipPolicy">
                <retryable-exception-classes>
                    <include class="org.springframework.dao.TransientDataAccessException" />
                </retryable-exception-classes>
```

Agenda

clearstream | DEUTSCHE BÖRSE GROUP

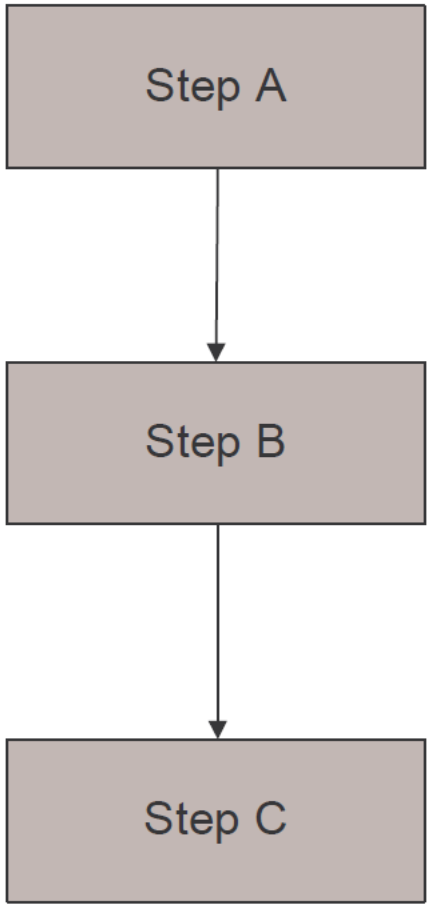
- Spring & Spring Batch Overview
- Domain Language of Batch
- Configuring and Running a Job
- Advanced Features
- Summary

Sequential Flow

clearstream | DEUTSCHE BÖRSE GROUP

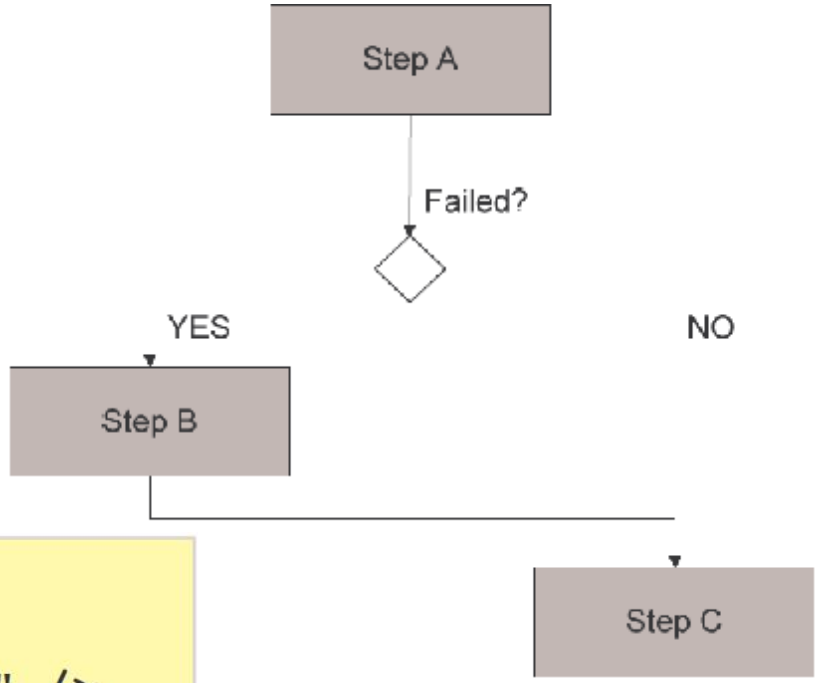
- The simplest flow scenario is a job where all of the steps execute sequentially
- This can be achieved using the 'next' attribute of the step element

```
<job id="job">
  <step id="stepA" next="stepB" />
  <step id="stepB" next="stepC"/>
  <step id="stepC"/>
</job>
```



Conditional Flow

– In order to handle more complex scenarios, Spring Batch allows transition elements to be defined within the step element

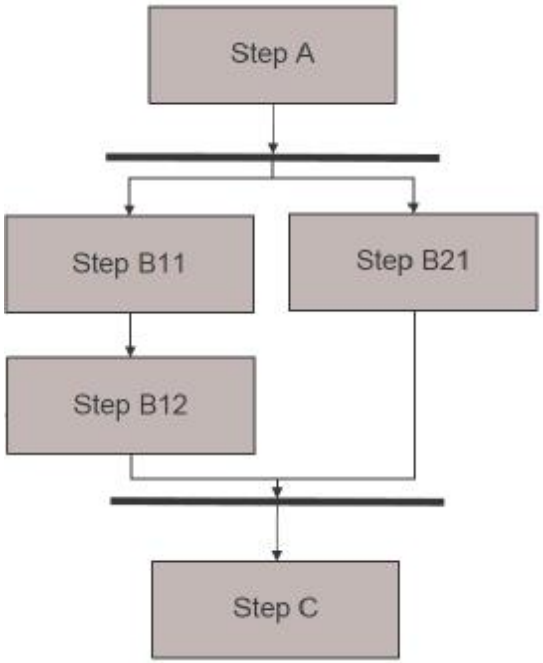



```
<job id="job">
  <step id="stepA">
    <next on="FAILED" to="stepB" />
    <next on="*" to="stepC" />
  </step>
  <step id="stepB" next="stepC" />
  <step id="stepC" />
</job>
```

Split Flow

– Spring Batch also allows for a job to be configured with parallel flows using the 'split' element

```
<split id="stepB" next="stepC">
  <flow>
    <step id="stepB11" next="stepB11"/>
    <step id="stepB12"/>
  </flow>
  <flow>
    <step id="stepB21"/>
  </flow>
</split>
<step id="stepC"/>
```



Skip & Retry & Fatal Logic

clearstream | DEUTSCHE BÖRSE GROUP

- Based on catching and processing Java exceptions
- Skip Logic
  - There are scenarios where errors encountered should not result in Step failure, but should be skipped instead
  - Avoid step fail when unique constrains violations, …
- Retry Logic
  - Ability to process one item more than once in order to avoid Step failure
  - Can handle optimistic locking exceptions, …
- Fatal Exceptions
  - Causes immediate failure when specified exception caught
  - Input file not found, …

Intercepting Job Execution

clearstream | DEUTSCHE BÖRSE GROUP

– You might need to perform some functionality at certain events during the execution of a Step
– Based on listeners

```
<step id="step1">
  <tasklet reader="reader" writer="writer"
           commit-interval="10"/>
 <listeners>
    <listener ref="stepListener"/>
 </listeners>
</step>
```

– StepExecutionListenerm, ChunkListener, ItemReadListener, ItemProcessListener, ItemWriteListener, SkipListener

clearstream | DEUTSCHE BÖRSE GROUP

Available Item Readers

Item Reader	Description
AbstractItemCountingItemStreamItemReader	Abstract base class that provides basic restart capabilities by counting the number of items returned from an `ItemReader`.
ListItemReader	Provides the items from a list, one at a time
ItemReaderAdapter	Adapts any class to the `ItemReader` interface.
AggregateItemReader	An ItemReader that delivers a list as its item, storing up objects from the injected ItemReader until they are ready to be packed out as a collection. This ItemReader should mark the beginning and end of records with the constant values in FieldSetMapper AggregateItemReader**#BEGIN_RECORD** and AggregateItemReader**#END_RECORD**
FlatFileItemReader	Reads from a flat file. Includes ItemStream and Skippable functionality. See section on Read from a File
StaxEventItemReader	Reads via StAX. See HOWTO - Read from a File
JdbcCursorItemReader	Reads from a database cursor via JDBC. See HOWTO - Read from a Database
HibernateCursorItemReader	Reads from a cursor based on an HQL query. See section on Reading from a Database
IbatisPagingItemReader	Reads via iBATIS based on a query. Pages through the rows so that large datasets can be read without running out of memory. See HOWTO - Read from a Database
JmsItemReader	Given a Spring JmsOperations object and a JMS Destination or destination name to send errors, provides items received through the injected JmsOperations receive() method
JpaPagingItemReader	Given a JPQL statement, pages through the rows, such that large datasets can be read without running out of memory
JdbcPagingItemReader	Given a SQL statement, pages through the rows, such that large datasets can be read without running out of memory

Available Item Writers

clearstream | DEUTSCHE BÖRSE GROUP

Item Writer	Description
AbstractItemStreamItemWriter	Abstract base class that combines the `ItemStream` and `ItemWriter` interfaces.
CompositeItemWriter	Passes an item to the process method of each in an injected **List** of **ItemWriter** objects
ItemWriterAdapter	Adapts any class to the `ItemWriter` interface.
PropertyExtractingDelegatingItemWriter	Extends AbstractMethodInvokingDelegator creating arguments on the fly. Arguments are created by retrieving the values from the fields in the item to be processed (via a SpringBeanWrapper) based on an injected array of field name
FlatFileItemWriter	Writes to a flat file. Includes ItemStream and Skippable functionality. See section on Writing to a File
HibernateItemWriter	This item writer is hibernate session aware and handles some transaction-related work that a non-"hibernate aware" item writer would not need to know about and then delegates to another item writer to do the actual writing.
JdbcBatchItemWriter	Uses batching freatures from a `PreparedStatement`, if available, and can take rudimentary steps to locate a failure during a `flush`.
JpaItemWriter	This item writer is JPA EntityManager aware and handles some transaction-related work that a non-"jpa aware" `ItemWriter` would not need to know about and then delegates to another writer to do the actual writing.
StaxEventItemWriter	Uses an **ObjectToXmlSerializer** implementation to convert each item to XML and then writes it to an XML file using StAX.

Agenda

clearstream | DEUTSCHE BÖRSE GROUP

- Spring & Spring Batch Overview
- Domain Language of Batch
- Configuring and Running a Job
- Advanced Features
- Summary

Summary

clearstream | DEUTSCHE BÖRSE GROUP

– Lack of a standard enterprise batch architecture is resulting in higher costs associated with the quality and delivery of solutions.

– Spring Batch provides a highly scalable, easy-to-use, customizable, industry-accepted batch framework

– Spring patterns and practices have been leveraged allowing developers to focus on business logic, while enterprise architects can customize and extend architecture concerns

Thank you!

clearstream | DEUTSCHE BÖRSE GROUP

– Questions ?


clearstream
DEUTSCHE BÖRSE
GROUP