

# Spring Batch

Lessons Learned out of a real life banking system.

Michael Beer  
*Senior Consultant*

Raffael Schmid  
*Consultant*



BASEL BERN BRUGG LAUSANNE ZUERICH DUESSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA

## ■ About us



**Michael Beer**  
*Senior Consultant*

- Trivadis since 2001
- design and development of web based applications
- part of the Trivadis APM team



**Raffael Schmid**  
*Consultant*

- Trivadis since 2010
- design and development of web based applications
- interested in performance related things on the JVM

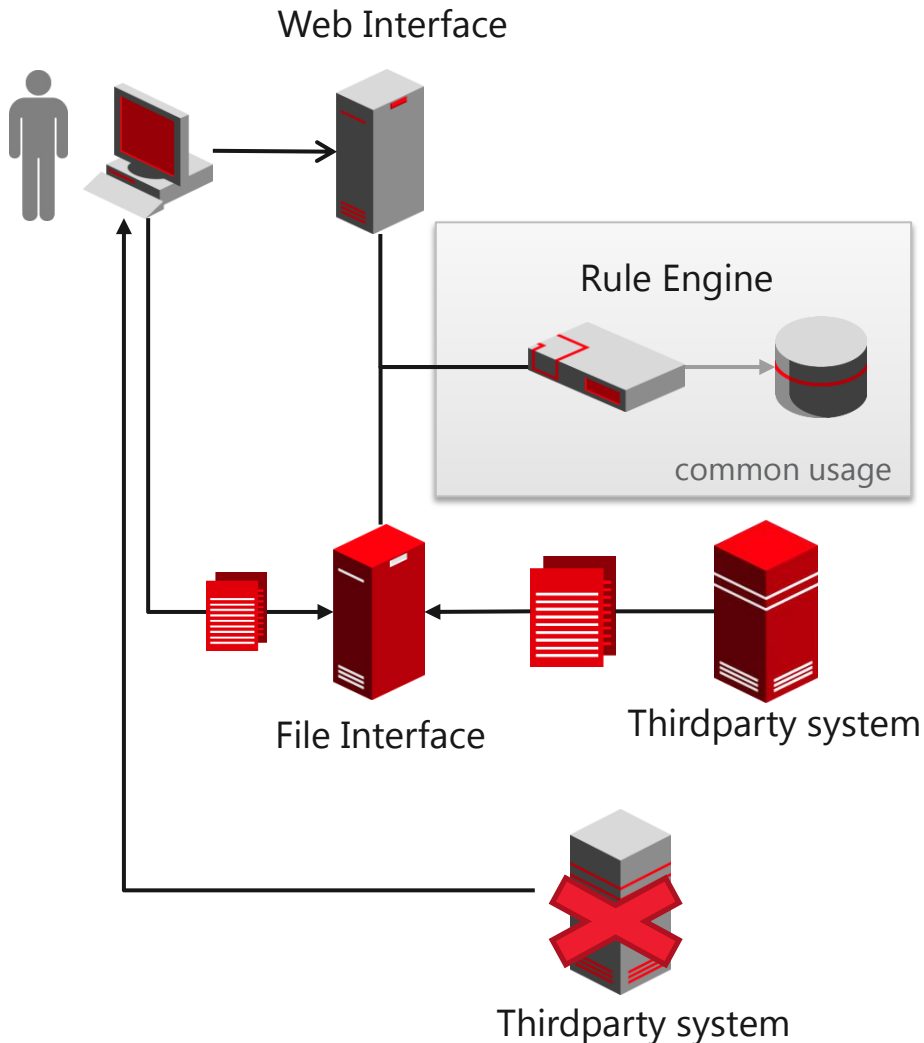
# ■ AGENDA



# ■ Initial position



# ■ Initial and target system context

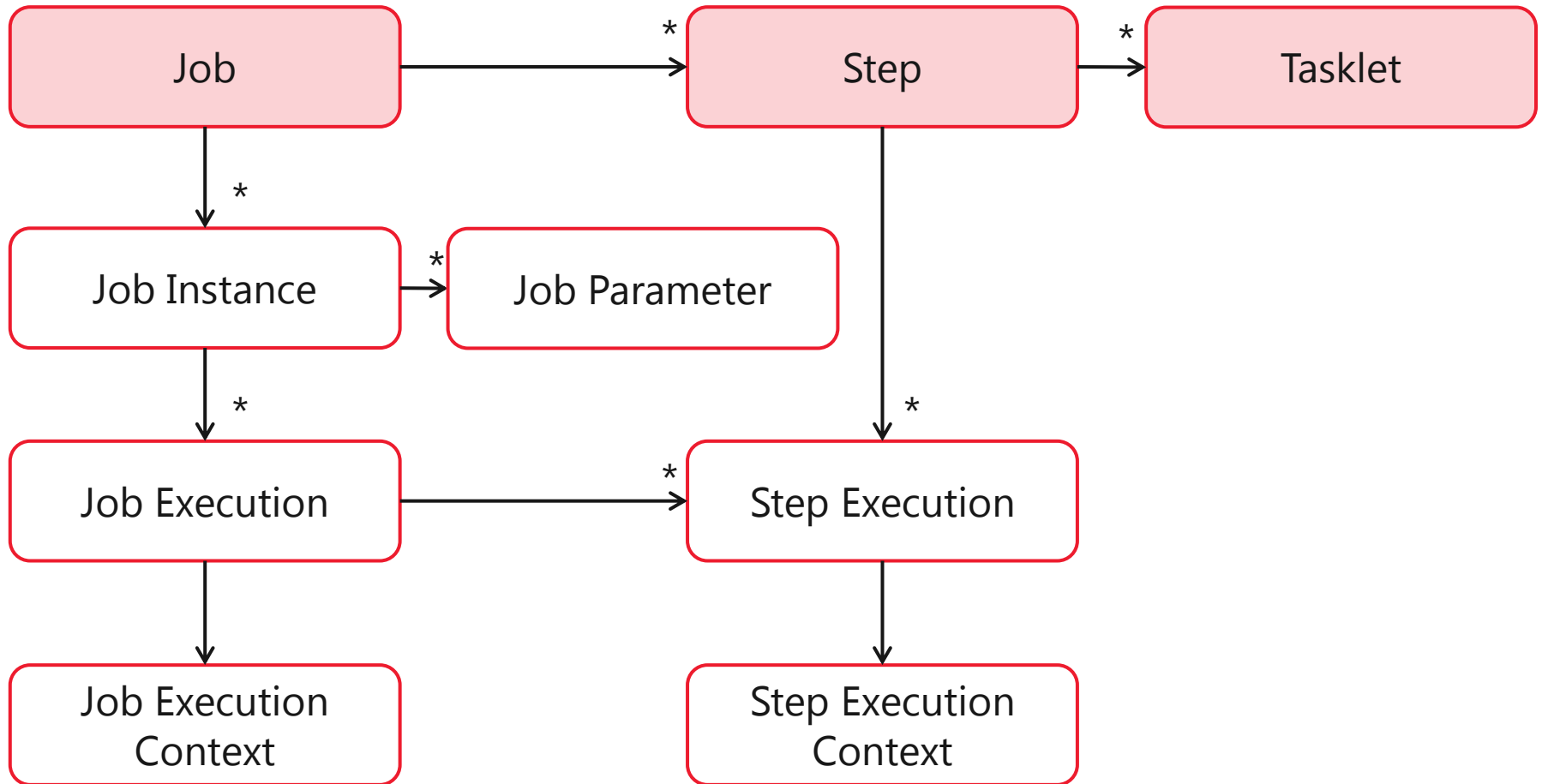


- Automatic load out of thirdparty systems
  - multiple times per day
  - export / import as CSV file
- Manual load by user
  - at irregular time intervals
- Migration load due to thirdparty system decommissioning
  - run once
  - different volumes

# ■ Why we used Spring Batch?

- It was already part of the technology stack within the customers environment.
- There were no real (free) alternatives at that time (or at least we didn't know any).
- Exceptional permit was needed for Spring Integration because it was shipped with Spring Batch Admin.

# Terminology of Spring Batch jobs



# ■ Requirements and Topics





## ■ Requirement: Performance

Performance

Reprocess  
failed items

Summary mail

Gather detailed  
job information

Trigger jobs

Control load

Deactivate jobs

Inter-job  
dependencies

- parallelized on multiple threads
- order of execution matters
- number of threads can be changed before or after job execution

## ■ Requirement: Reprocess failed items

Performance

Reprocess  
failed items

Summary mail

Gather detailed  
job information

Trigger jobs

Control load

Deactivate jobs

Inter-job  
dependencies

- rerunning a job processes failed items (only)
- process state of item therefore needs to be maintained

## ■ Requirement: Summary mail

Performance

Reprocess  
failed items

Summary mail

Gather detailed  
job information

Trigger jobs

Control load

Deactivate jobs

Inter-job  
dependencies

- write a detailed execution log that contains
  - Exceptions
  - messages out of the Rule Engine
- aggregate to summary
  - Number of errors, warnings, etc.
  - worst status level
- send summary to submitter and operator

## ■ Requirement: Gather detailed job information

Performance

Reprocess  
failed items

Summary mail

Gather detailed  
job information

Trigger jobs

Control load

Deactivate jobs

Inter-job  
dependencies

- rerun failed single item in trace mode
- collect diagnostic information
  - e.g. out of the Rule Engine

## ■ Requirement: Trigger jobs

Performance

Reprocess  
failed items

Summary mail

Gather detailed  
job information

Trigger jobs

Control load

Deactivate jobs

Inter-job  
dependencies

- periodically
  - either fixed delay or fixed rate
- on event
  - e.g. new data arrived in database

## ■ Requirement: Control load

Performance

Reprocess  
failed items

Summary mail

Gather detailed  
job information

Trigger jobs

Control load

Deactivate jobs

Inter-job  
dependencies

- prevent too many jobs in parallel
- conditions might prevent job execution
- requests should prevent system shutdown

## ■ Requirement: Deactivate jobs

Performance

Reprocess  
failed items

Summary mail

Gather detailed  
job information

Trigger jobs

Control load

Deactivate jobs

Inter-job  
dependencies

- deactivate job execution
- set job execution on hold

## ■ Requirement: Inter-job dependencies

Performance

Reprocess  
failed items

Summary mail

Gather detailed  
job information

Trigger jobs

Control load

Deactivate jobs

Inter-job  
dependencies

- finished jobs might trigger dependencies



## ■ Requirements grouped into five different topics

Partitioning

Error  
Handling

Monitoring & Tracing

Trigger jobs

Job Control

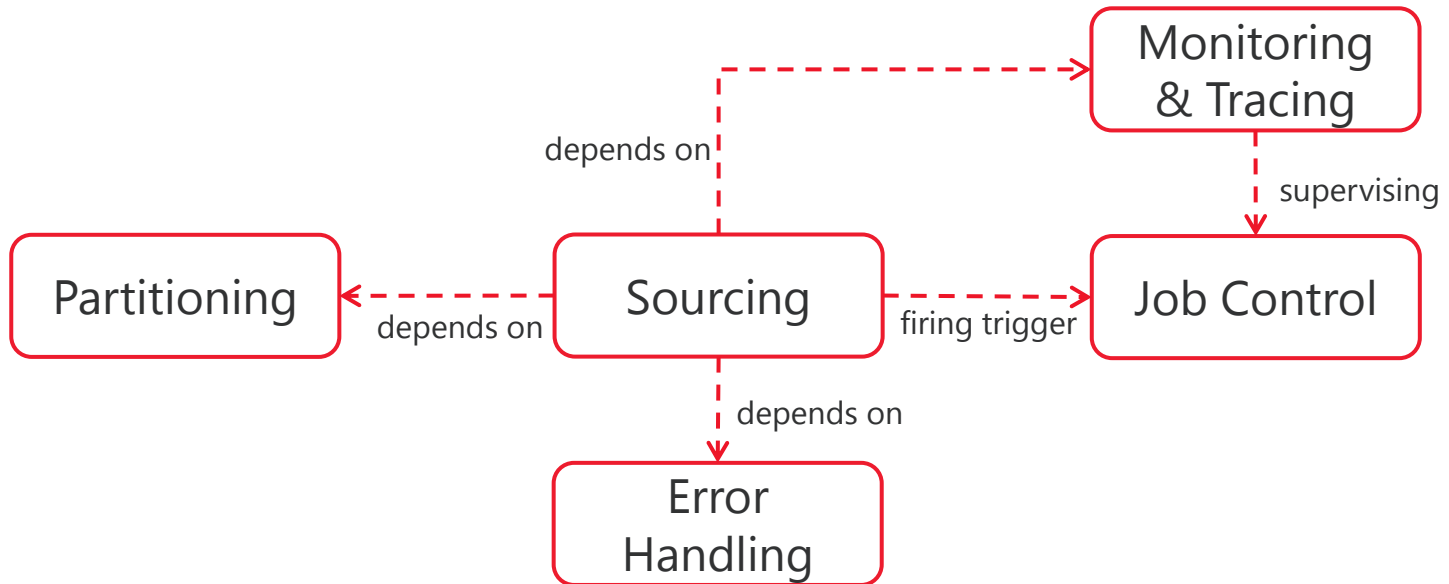
Inter-job  
dependencies

Sourcing

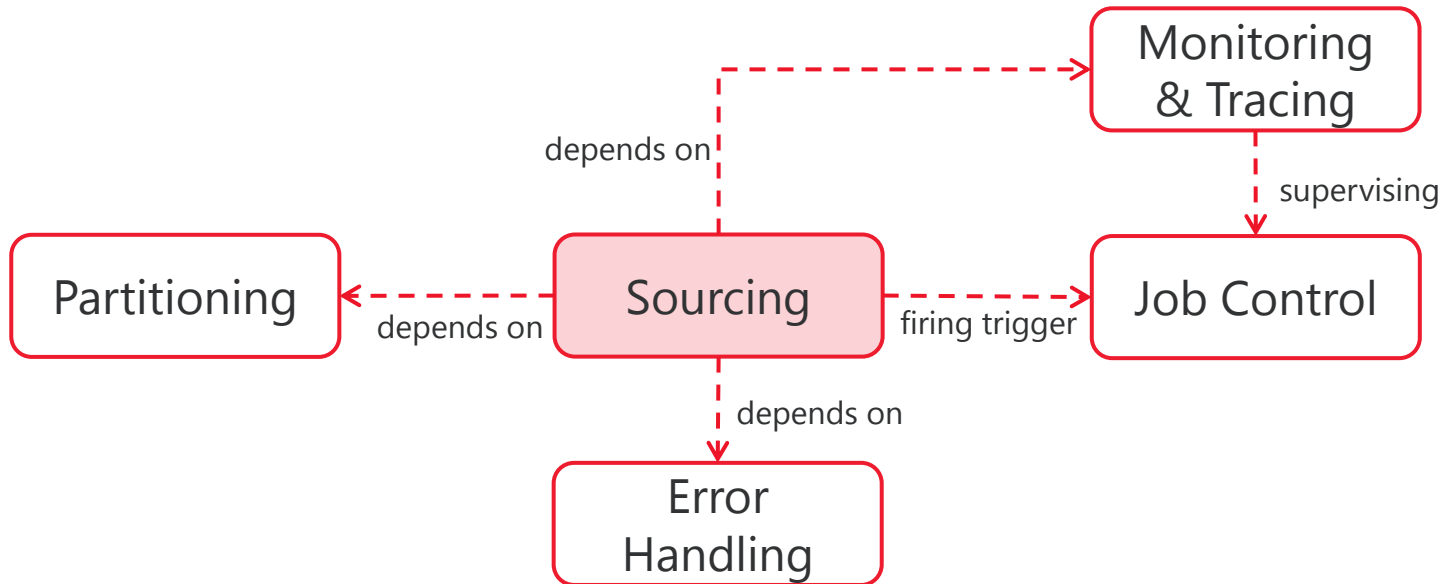
# ■ Lessons learned



# ■ Topics we will cover



# ■ Sourcing



## ■ Load data into staging table

CAT_ID	PERSON_ID	AMOUNT	CURRENCY	VALID_FORM	LAST_CHANGE
310	2908	200000	USD	03.12.2013	02.02.2013
310	1608	100000	CHF	03.12.2013	02.01.2013
311	1608	100000	CHF	03.12.2013	02.01.2013

PARTITION	LINE	TYPE	STATE	FIELD01	FIELD02	FIELD03	FIELD04
PT_000122	171	D	02	441	1804	400000	EUR
PT_000122	172	D	02	441	1002	221000	EUR
PT_000123	1	H	01	CAT_ID	PERSON_ID	AMOUNT	CURRENCY
PT_000123	2	D	01	310	2908	200000	USD
PT_000123	3	D	01	310	1608	100000	CHF
PT_000123	4	D	01	311	1608	100000	CHF

## ■ Allows partitioning, single record execution

PARTITION	LINE	TYPE	STATE	FIELD01	FIELD02	FIELD03	FIELD04
PT_000123	1	H	01	CAT_ID	PERSON_ID	AMOUNT	CURRENCY
PT_000123	2	D	01	310	2908	200000	USD
PT_000123	3	D	01	310	1410	100000	CHF

### ■ Partitioning out of the box

```
SELECT * FROM LOAD
      WHERE PARTITION = 'PT_000123'
      AND LINE BETWEEN 0 AND 99;
```

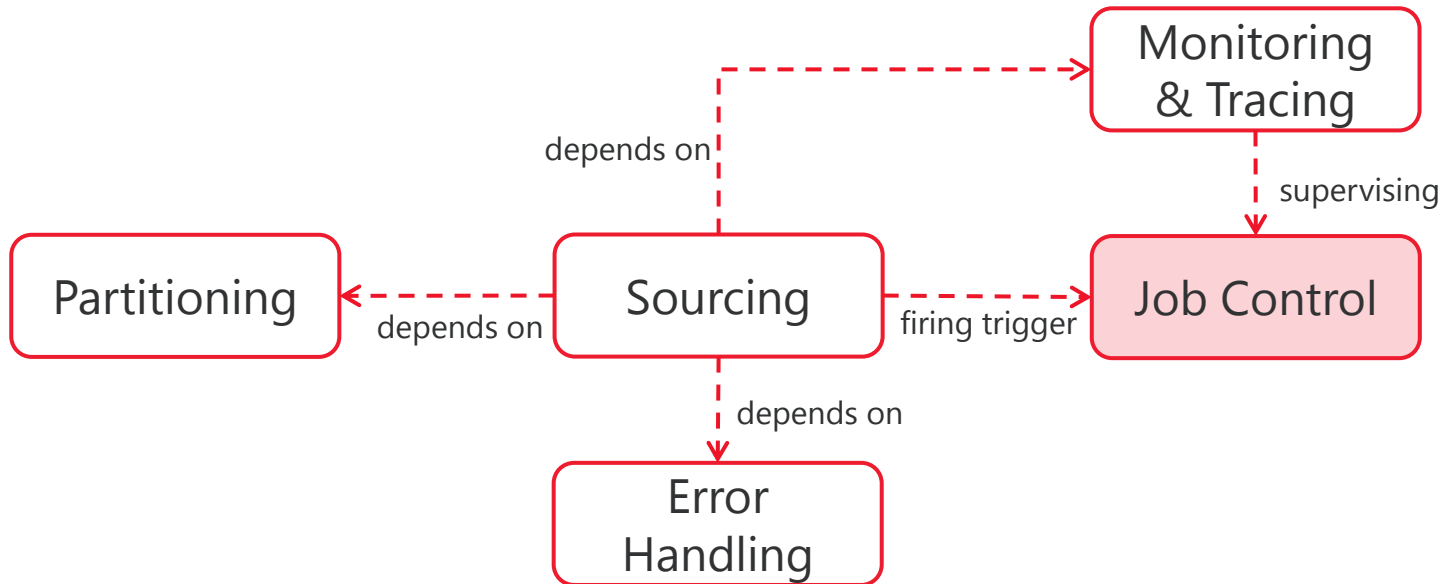
### ■ Reexecution of failed records

```
SELECT * FROM LOAD
      WHERE PARTITION = 'PT_000123'
      AND STATE = '01'
```

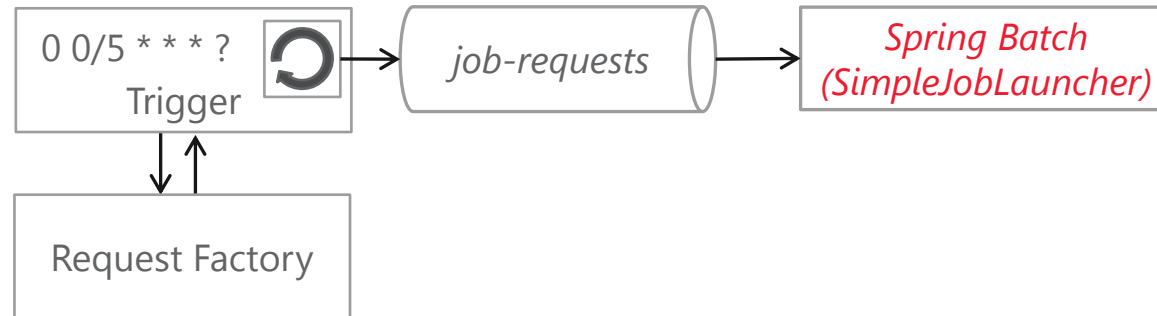
### ■ Execution of single record

```
SELECT * FROM LOAD
      WHERE PARTITION = 'PT_000123'
      AND LINE = 8
```

# ■ Job Control



## ■ Simplest way of triggering a job



- Define polling channel adapter (CronTrigger)

```

<int:inbound-channel-adapter method="create" channel="job-requests"
ref="requestFactory">
  <int:poller cron="0 0/5 * * * ?" />
</int:inbound-channel-adapter>
  
```

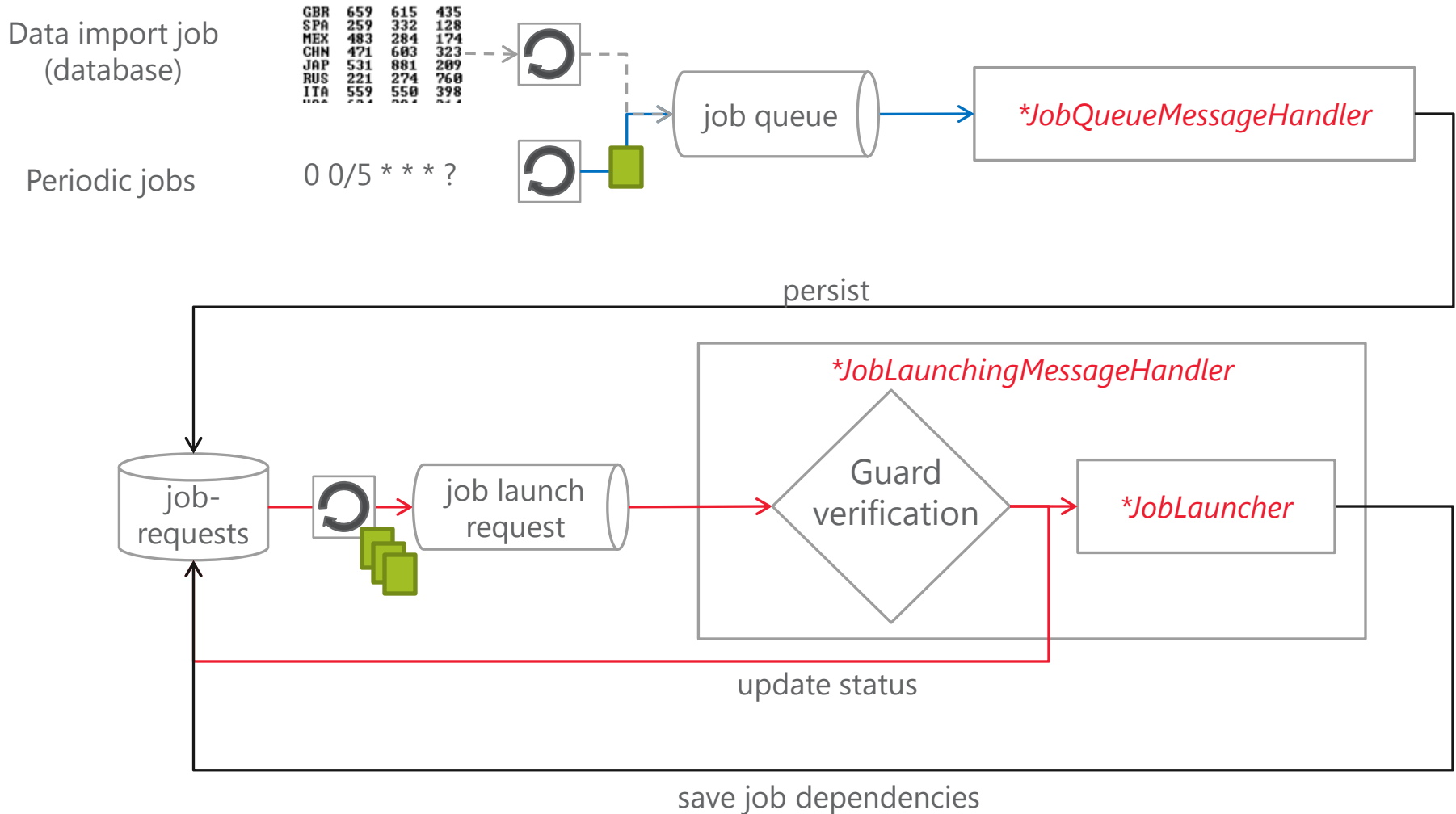
- Or even simpler (PeriodicTrigger)

```

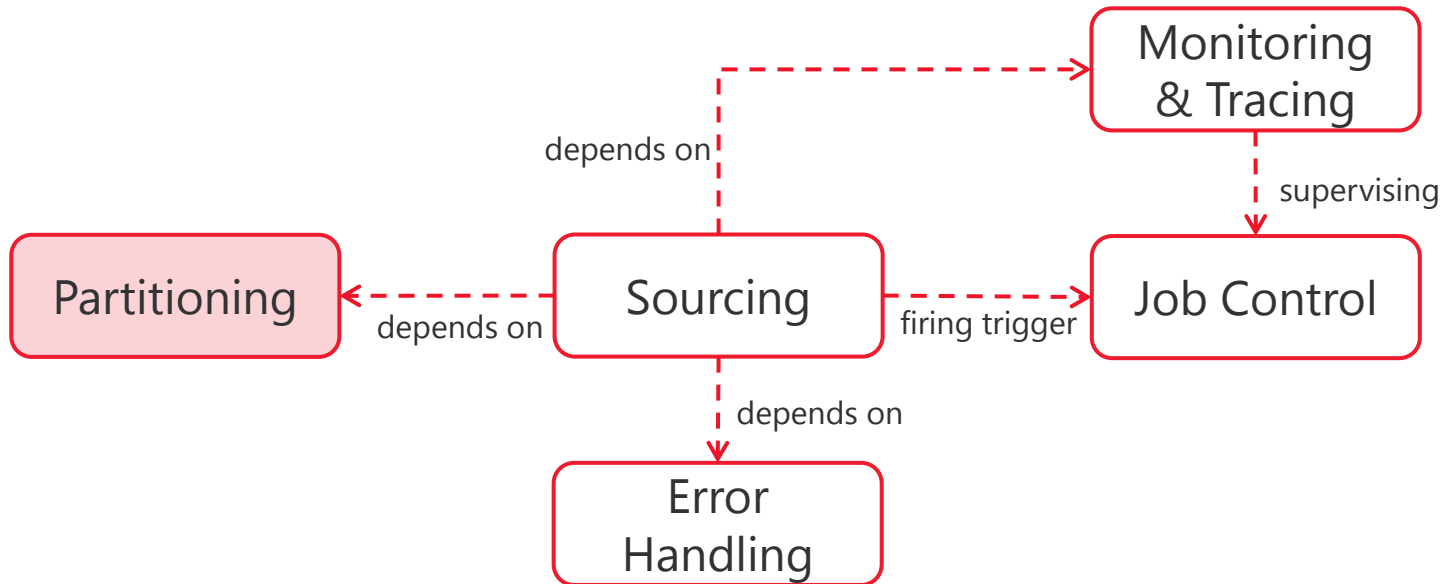
<int:inbound-channel-adapter method="create" channel="job-requests"
ref="requestFactory">
  <int:poller fixed-rate="300000"/>
</int:inbound-channel-adapter>
  
```



# Persist job launch requests into database



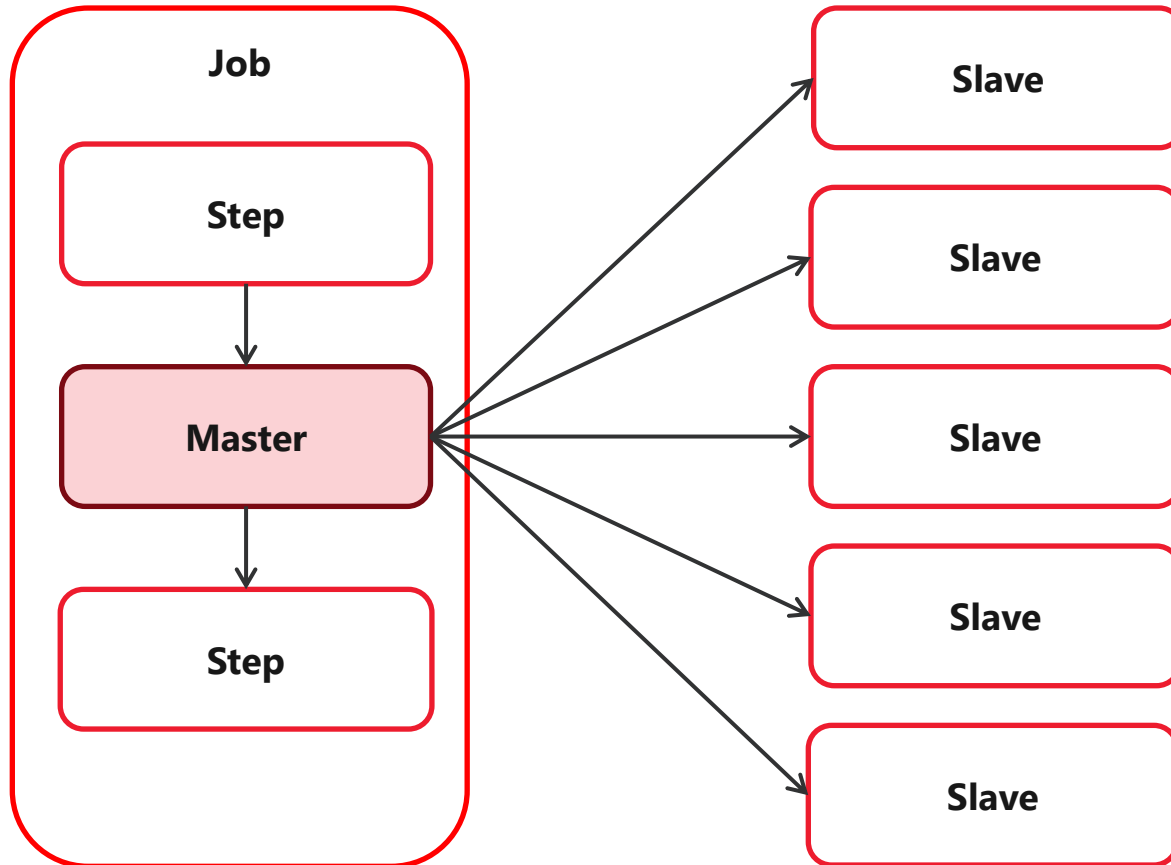
# ■ Partitioning



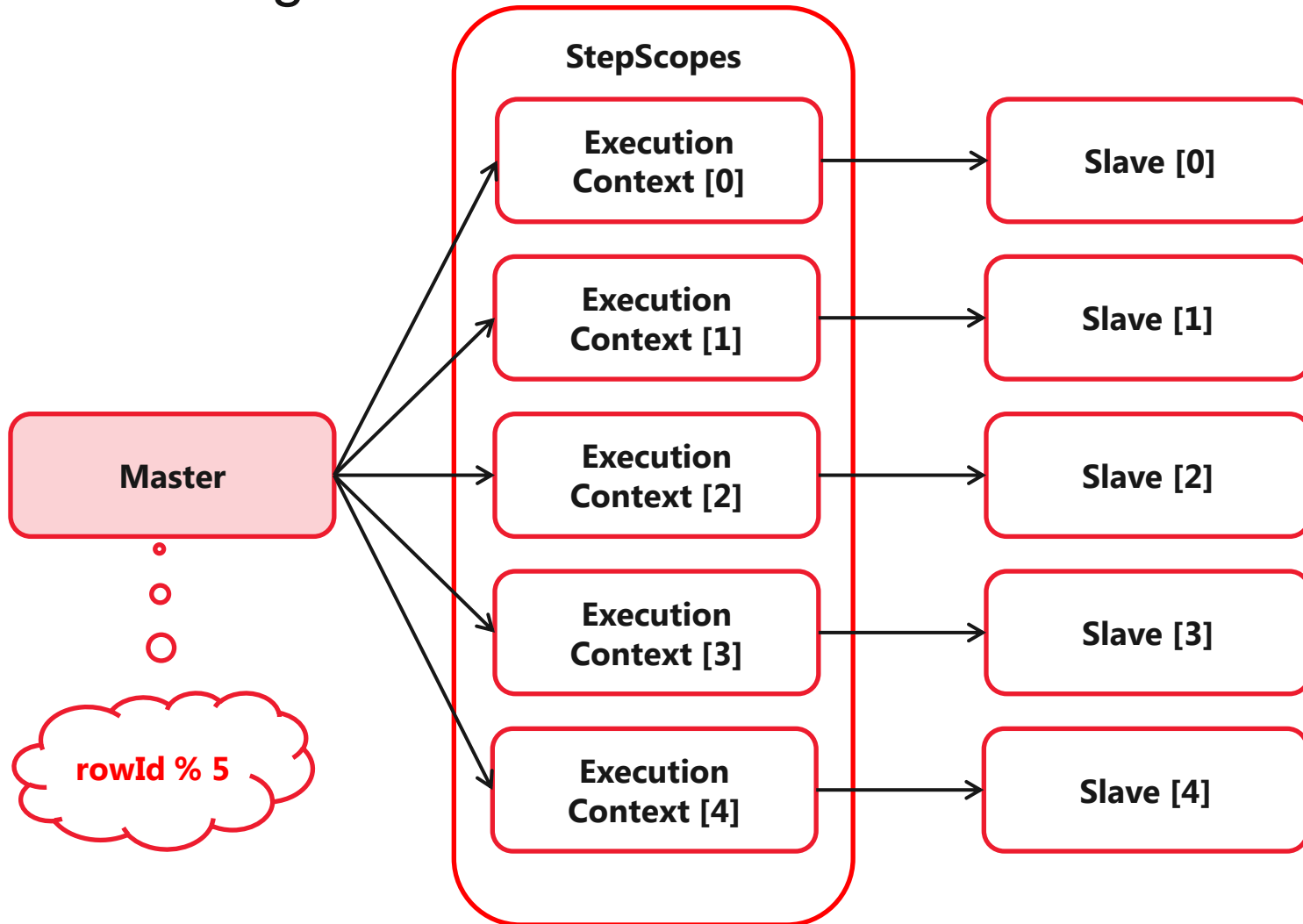
# ■ Performance

Type	Local/Remote	Description
<b>Multi-threaded Step</b>	Local	A step is multithreaded (TaskExecutor)
<b>Parallel Steps</b>	Local	Executes steps in parallel using multithreading
<b>Partitioning Step</b>	Local Remote	Partitions data and splits up processing
<b>Remote Chunking</b>	Remote	Distributed chunk processing to remote nodes

# ■ Partitioning overview



# ■ Partitioning detail



## ■ Characteristics of data

There might be **dependencies between records**. The order of execution matters at some point.

CAT_ID	PERSON_ID	AMOUNT	CURRENCY	VALID_FORM	LAST_CHANGE
310	2908			03.12.2013	02.02.2013
310	1608	100000	CHF	03.12.2013	02.01.2013
311	1608	100000	CHF	03.12.2013	02.01.2013
312	1608	100000	CHF	03.12.2013	02.01.2013
313	1608	100000	CHF	03.12.2013	02.01.2013
310	1410	100000	CHF	03.12.2013	02.01.2013
390	1108	100000	CHF	03.12.2013	02.01.2013
310	2908			04.12.2013	03.02.2013

Annotations: A grey arrow points from the empty AMOUNT cell in the first row to the word "Insert" in a grey box. A red arrow points from the empty AMOUNT cell in the last row to the word "Update" in a red box.

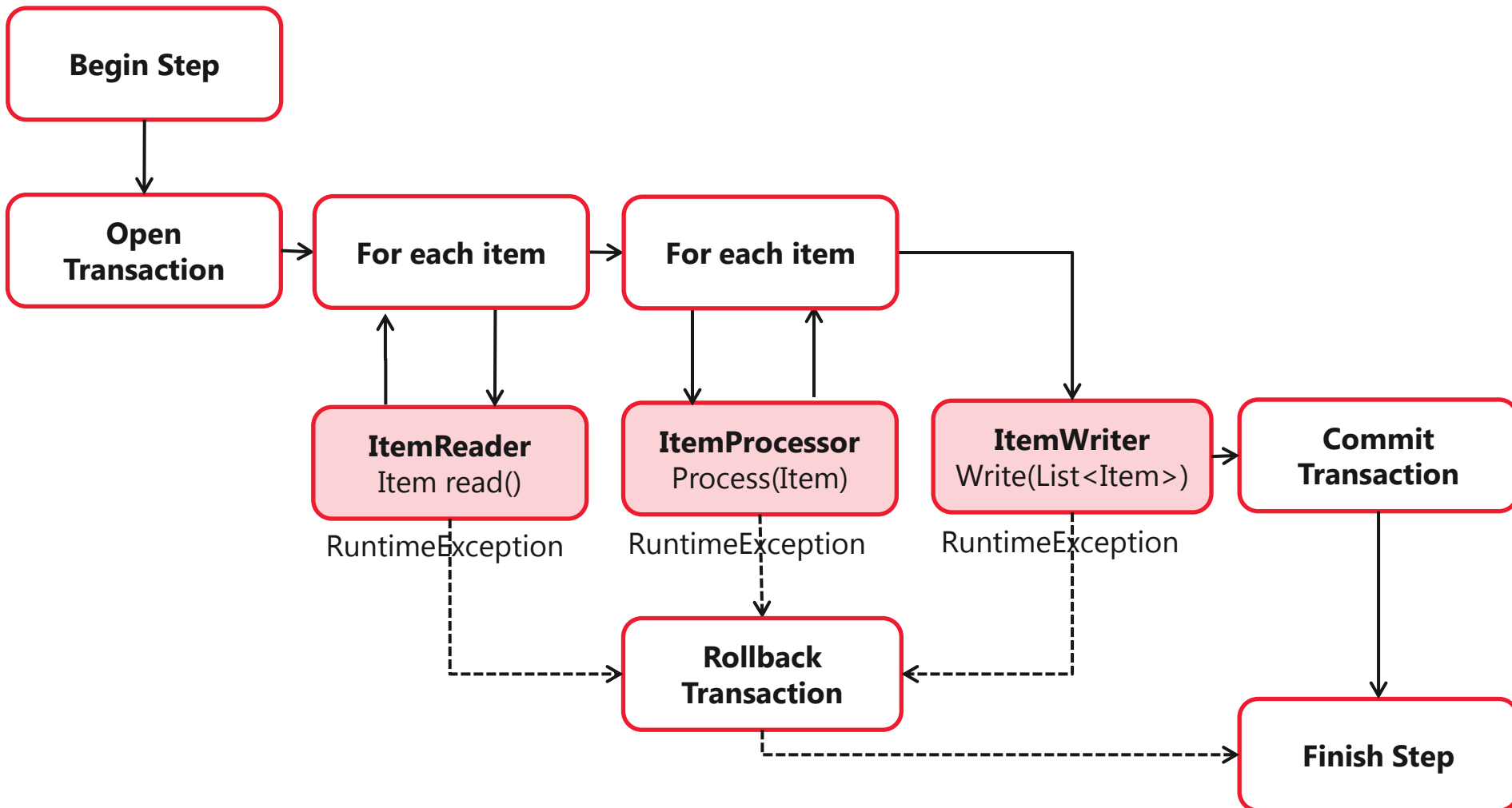


## ■ Partitioning detail – Spring Batch Admin

Property	Value
ID	0
Job Name	<u>csv-partition-sample-job</u>
Job Instance	<u>0</u>
Job Parameters	1=2
Start Date	2013-09-25
Start Time	13:50:45
Duration	00:00:00
Status	COMPLETED
Exit Code	COMPLETED
Step Executions Count	<u>6</u>

StepName	Reads	Writes	Commits	Rollbacks	Duration	Status
partitionMaster	20	20	15	0	00:00:00	<u>COMPLETED</u>
partitionSlave:partition3	4	4	3	0	00:00:00	<u>COMPLETED</u>
partitionSlave:partition2	4	4	3	0	00:00:00	<u>COMPLETED</u>
partitionSlave:partition4	4	4	3	0	00:00:00	<u>COMPLETED</u>
partitionSlave:partition1	4	4	3	0	00:00:00	<u>COMPLETED</u>
partitionSlave:partition0	4	4	3	0	00:00:00	<u>COMPLETED</u>

# ■ Performance – Read / Write / Process





# ■ Performance

No error

Property	Min	Max	Mean	Sigma
Duration	22,957	22,957	22,957	0
Commits	101	101	101	0
Rollbacks	0	0	0	0
Reads	1,000	1,000	1,000	0
Writes	1,000	1,000	1,000	0
Filters	0	0	0	0
Read Skips	0	0	0	0
Write Skips	0	0	0	0
Process Skips	0	0	0	0

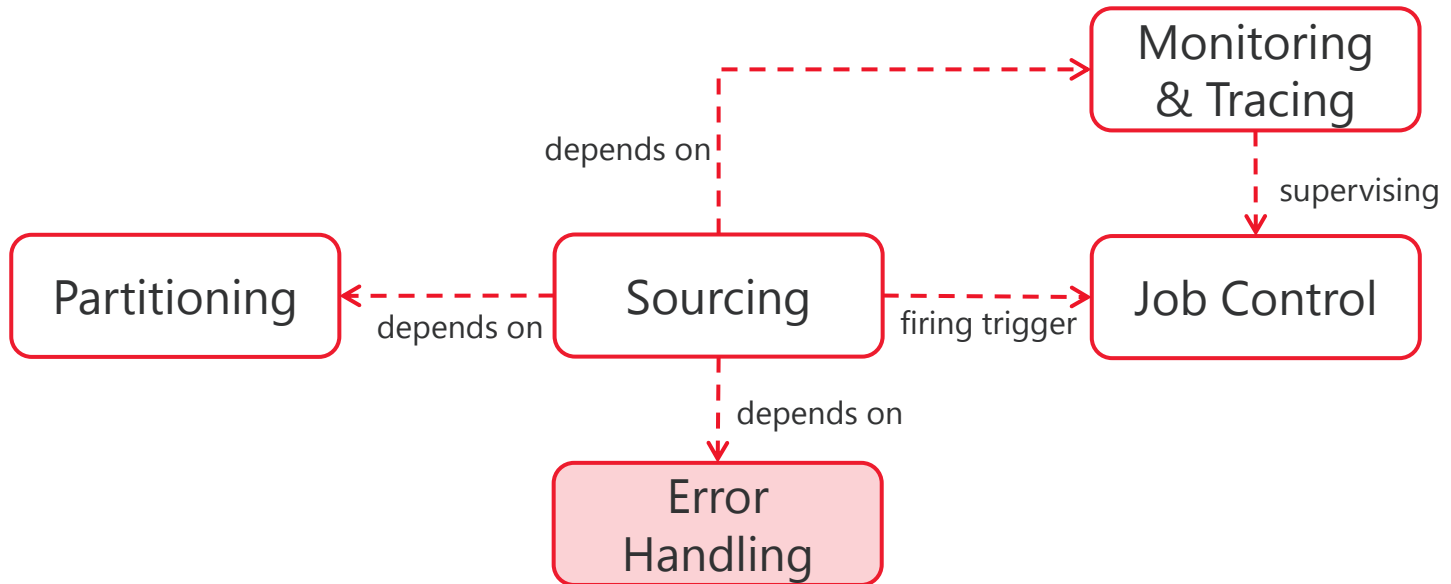
- ~ 22 sec
- 0 Rollbacks

Error on each item

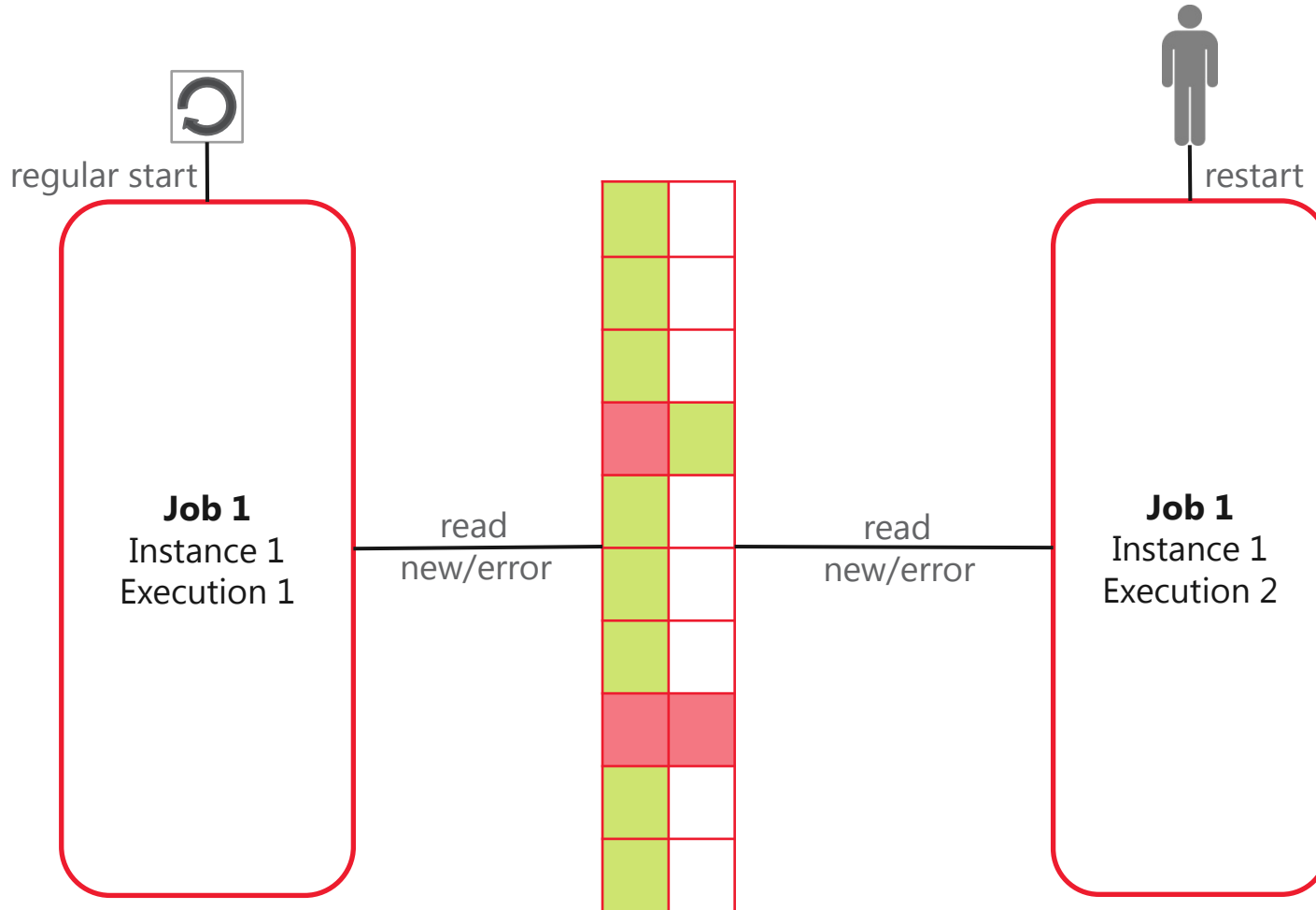
Property	Min	Max	Mean	Sigma
Duration	50,331	50,331	50,331	0
Commits	101	101	101	0
Rollbacks	1,100	1,100	1,100	0
Reads	1,000	1,000	1,000	0
Writes	0	0	0	0
Filters	5,500	5,500	5,500	0
Read Skips	0	0	0	0
Write Skips	1,000	1,000	1,000	0
Process Skips	0	0	0	0

- ~ 50 sec
- 1'100 Rollbacks
- 5'500 Filter
- 1'000 Write Skips

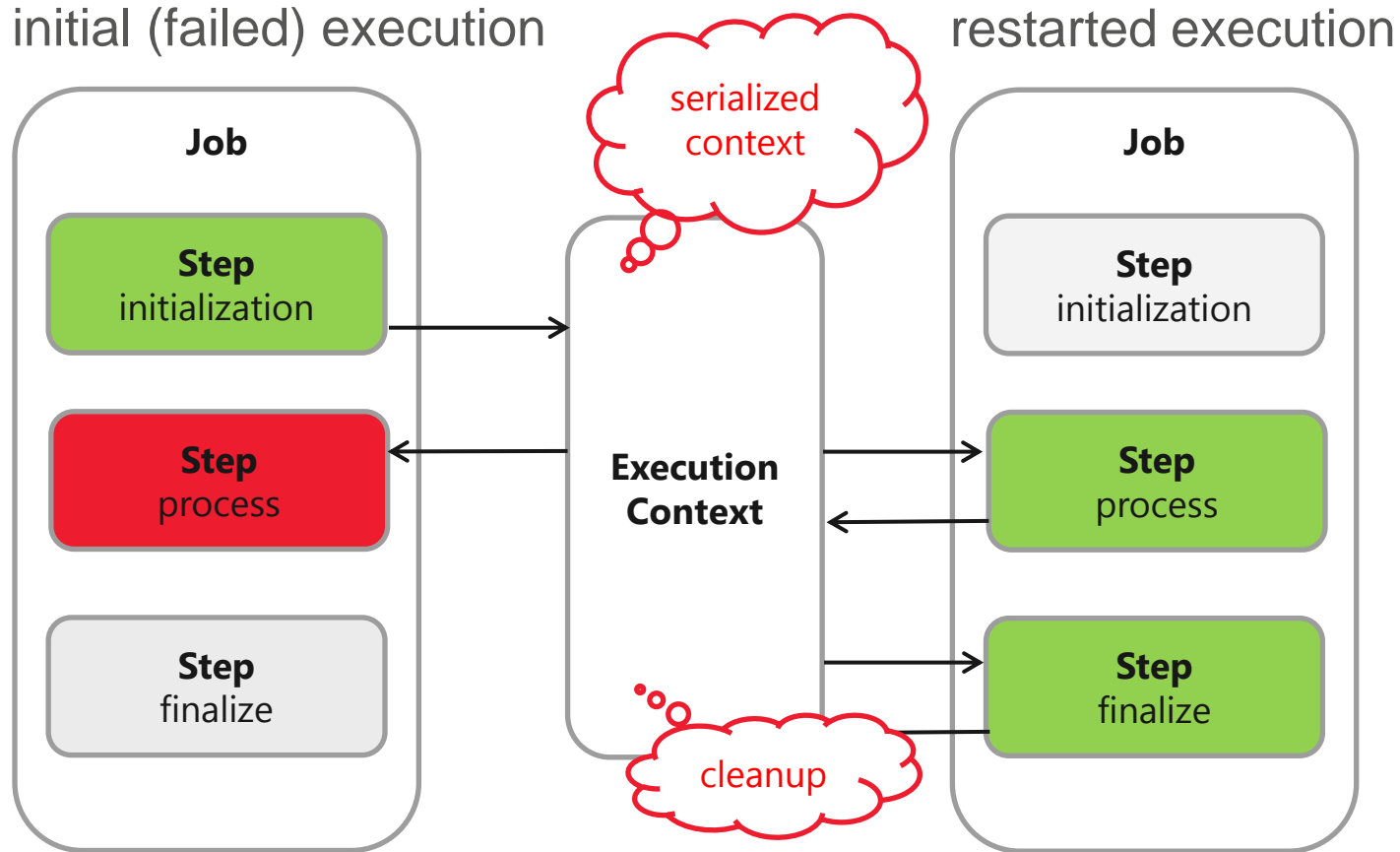
# ■ Error Handling



# Restartability



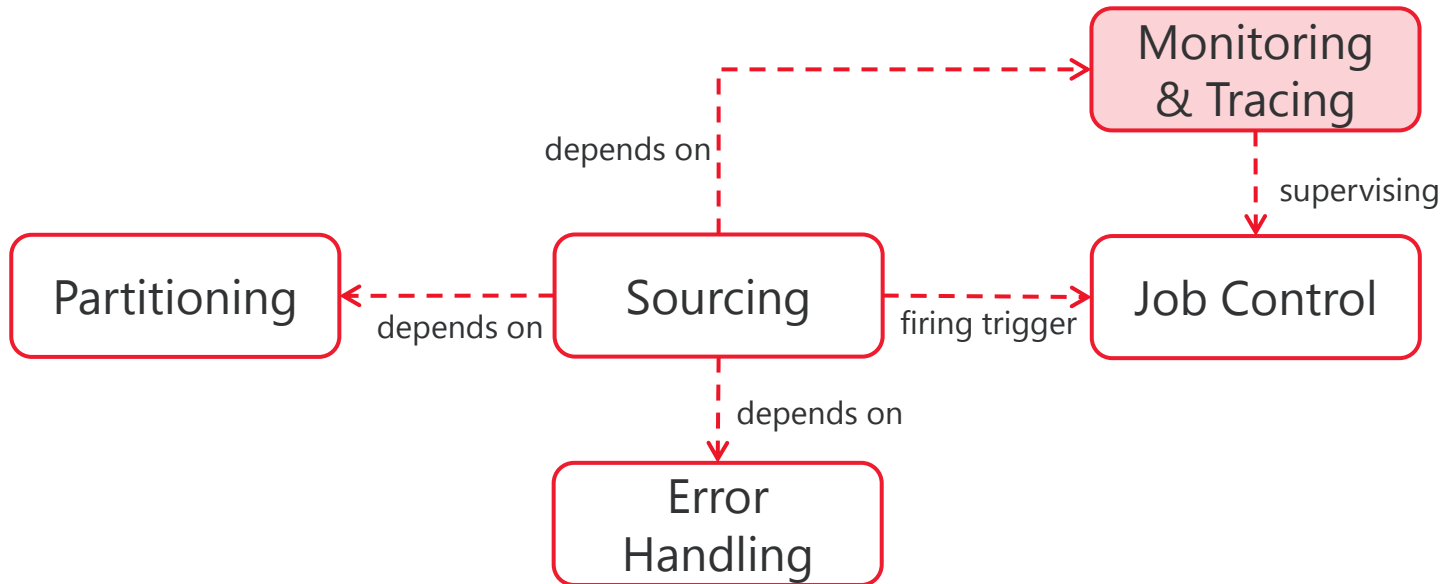
# Restartability configuration



## ■ Item based error handling

Feature	When?	What?	Where?
<b>Skip</b>	For nonfatal exceptions	Keeps processing for an incorrect item	Chunk-oriented step
<b>Retry</b>	For transient exceptions	Makes new attempts on an operation	Chunk-oriented step, application code

# ■ Monitoring



# ■ Monitoring – Spring Batch Admin

## Spring Batch Admin

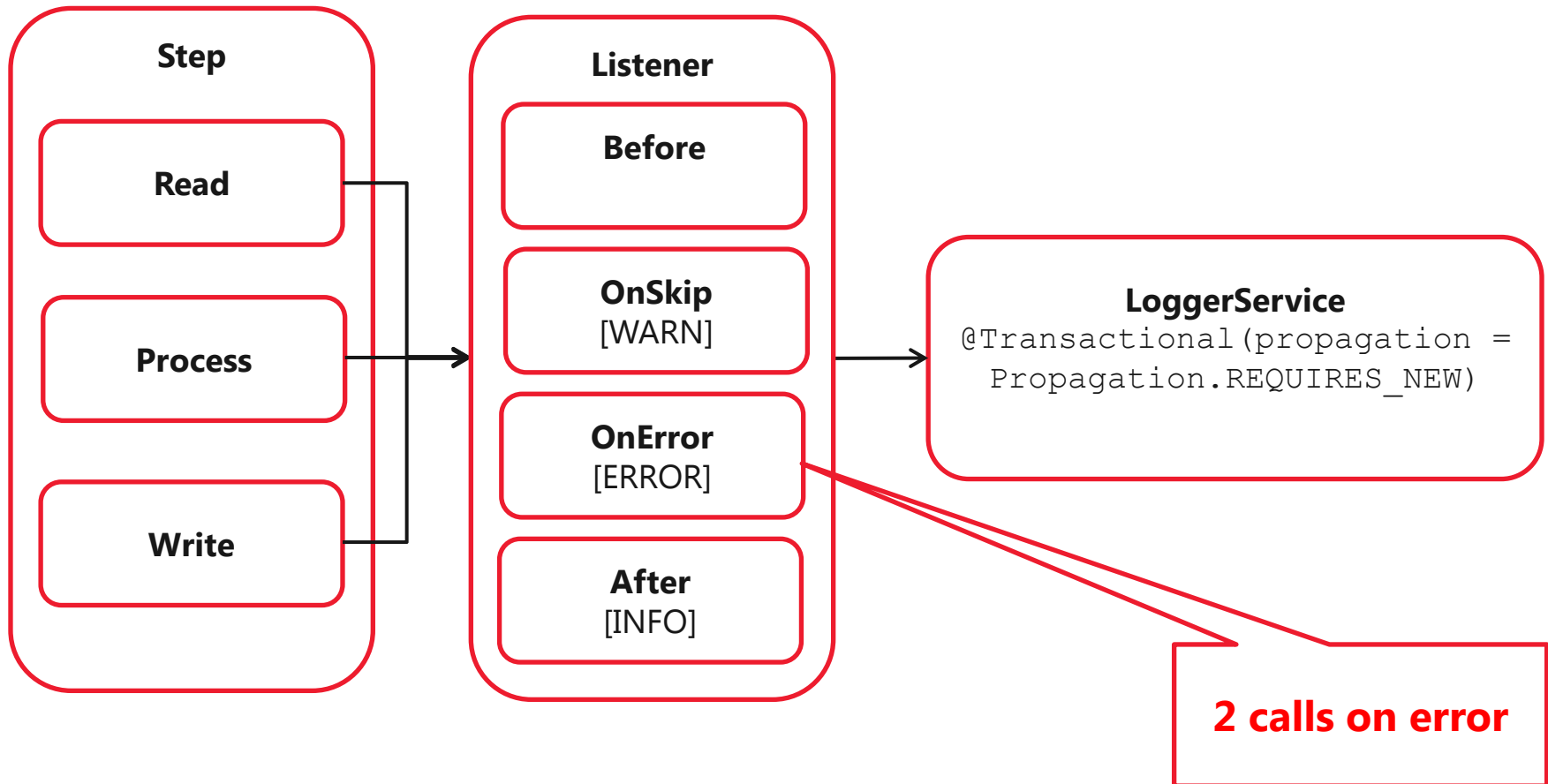


Exit Message  
 org.springframework.batch.item.ItemStreamException: Failed to initialize the reader at  
 org.springframework.batch.item.support.AbstractItemCountingItemStreamItemReader.open(AbstractItemCountingItemStreamItemReader.java:137) at  
 sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57) at  
 sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.lang.reflect.Method.invoke(Method.java:606) at  
 org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:309) at  
 org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:183) at  
 org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:150) at  
 org.springframework.aop.support.DelegatingIntroductionInterceptor.doProceed(DelegatingIntroductionInterceptor.java:131) at  
 org.springframework.aop.support.DelegatingIntroductionInterceptor.invoke(DelegatingIntroductionInterceptor.java:116)

Caused by: [java.lang.IllegalStateException](#): Input resource must exist (reader is in 'strict' mode): class path resource [nixDa]  
 at org.springframework.batch.item.file.FlatFileItemReader.doOpen(FlatFileItemReader.java:250)  
 at org.springframework.batch.item.support.AbstractItemCountingItemStreamItemReader.open(AbstractItemCountingItemStreamItemReader.java:134)

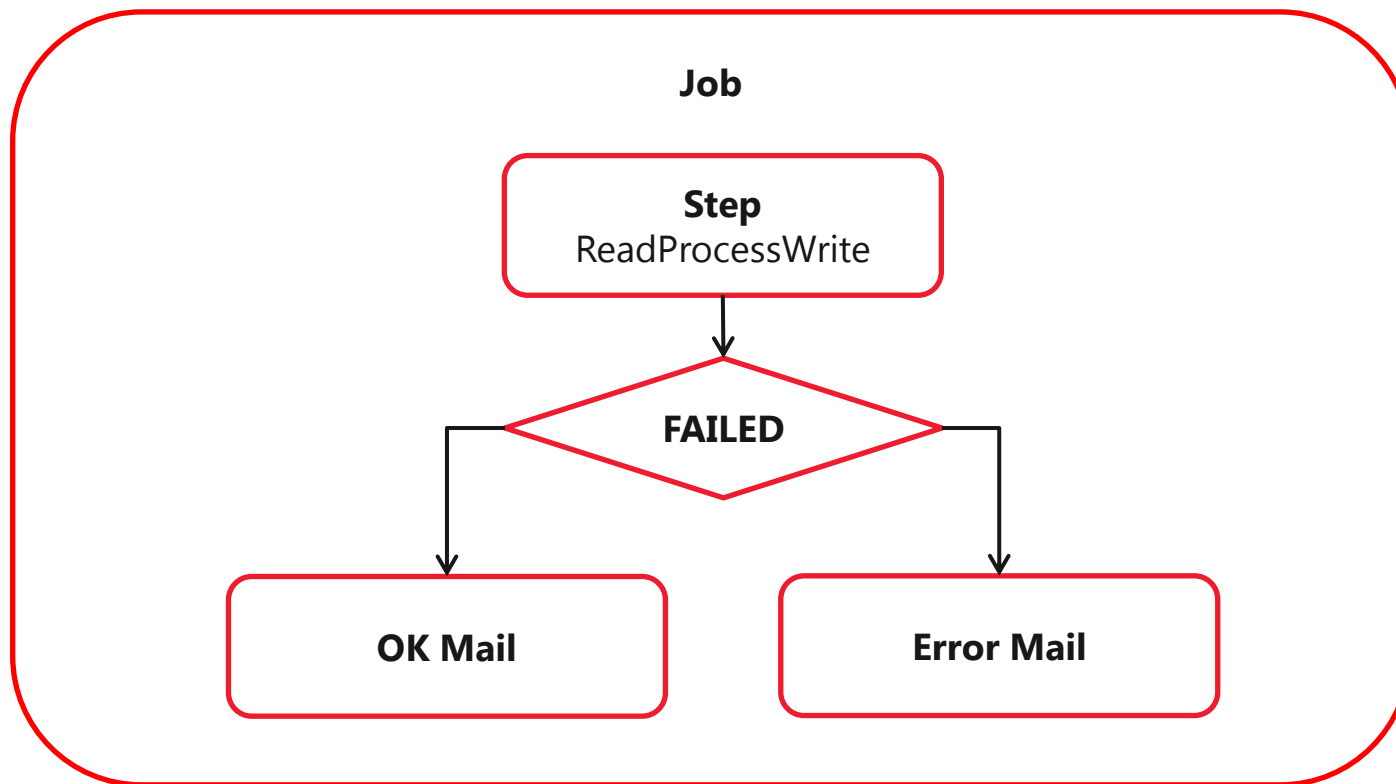
org.springframework.batch.core.step.tasklet.TaskletStep.open(TaskletStep.java:301) at  
 org.springframework.batch.core.step.AbstractStep.execute(AbstractStep.java:192) at  
 org.springframework.batch.core.job.SimpleStepHandler.handleStep(SimpleStepHandler.java:135) at  
 org.springframework.batch.core.job.flow.JobFlowExecutor.executeStep(JobFlowExecutor.java:61) at  
 org.springframework.batch.core.job.flow.support.state.StepState.handle(StepState.java:60) at  
 org.springframework.batch.core.job.flow.support.SimpleFlow.resume(SimpleFlow.java:144) at  
 org.springframework.batch.core.job.flow.support.SimpleFlow.start(SimpleFlow.java:124) at  
 org.springframework.batch.core.job.flow.FlowJob.doExecute(FlowJob.java:135) at  
 org.springframework.batch.core.job.AbstractJob.execute(AbstractJob.java:281) at  
 org.springframework.batch.core.launch.support.SimpleJobLauncher\$1.run(SimpleJobLauncher.java:120) at  
 java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145) at  
 java.util.concurrent.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:615)

# ■ Monitoring – logging to a database

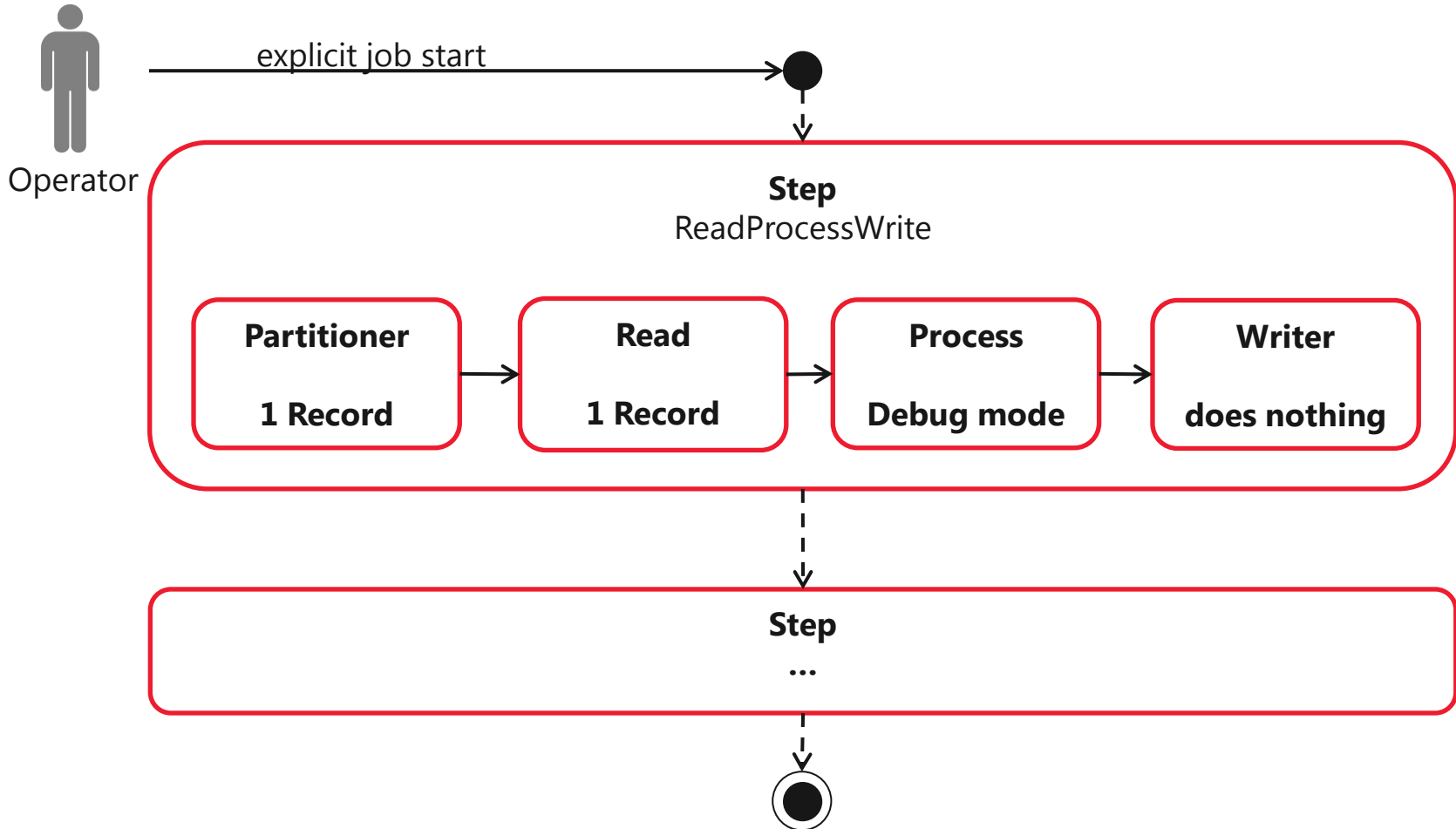




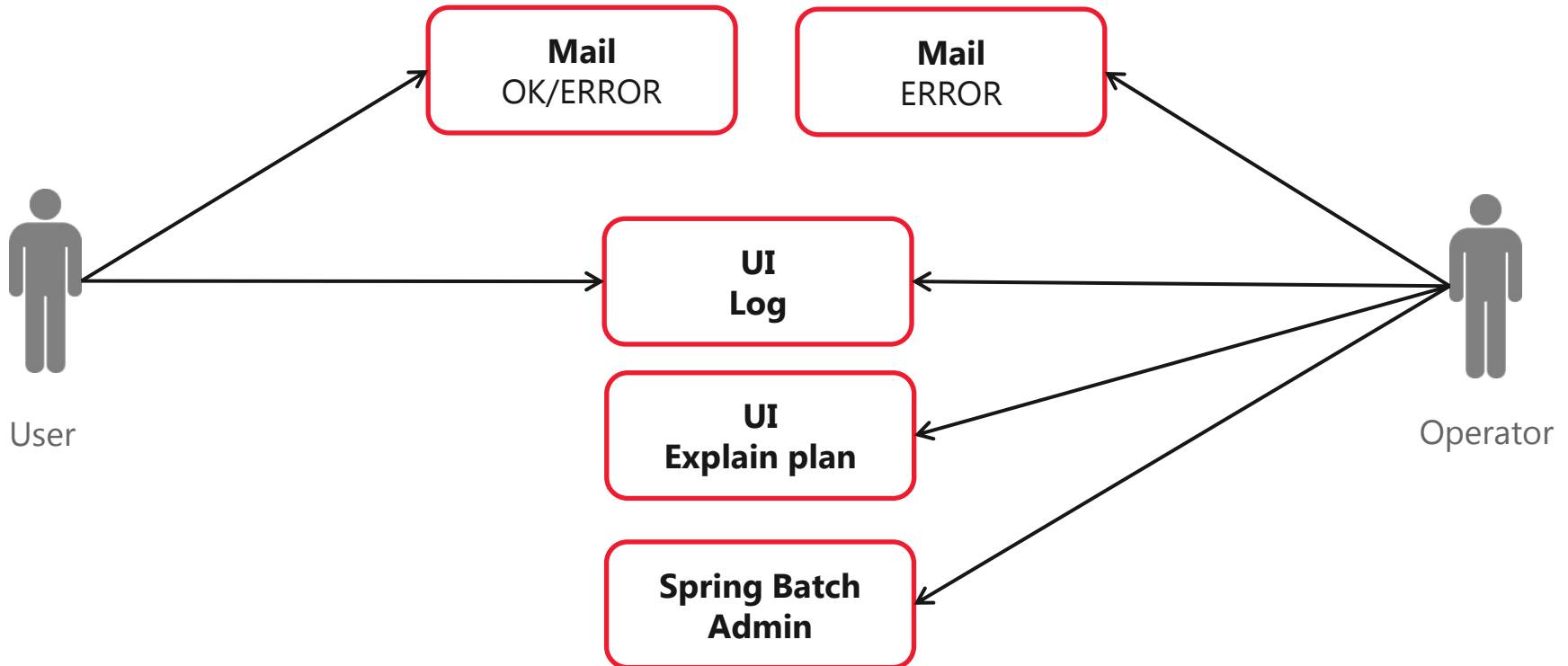
# ■ Monitoring - Mail



# ■ Tracing – Explain plan



# ■ Monitoring / Tracing summary



# ■ Conclusion



# ■ Challenges – Error Handling

- Fire and forget
  - Commit until first error
    - Skip
    - Retry
- Depending on the characteristic of data
  - Rollback
    - Whole job
    - Partition
    - Chunk
- Job flow
  - Deciders
- Operator takes care about the errors

# ■ Why we still use Spring Batch

Out of 155 SE tests in the JSR-352 TCK, Spring Batch 3.0 (milestone 1) passes 70.

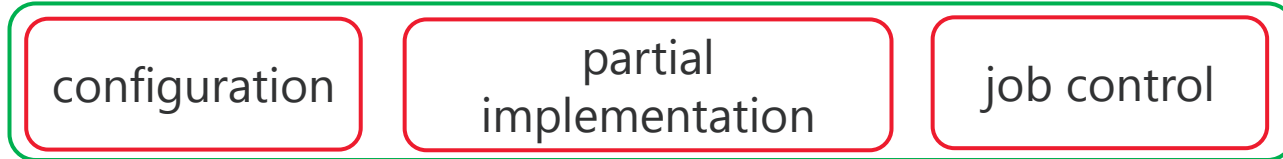
The terminology stays more or less the same: Job, Step, Chunk, Item, ItemProcessor, JobInstance, JobExecution.

Spring Batch	JSR 352	Comments
Tasklet	<b>Batchlet</b>	
ItemReader / ItemStream	<b>ItemReader</b>	JSR-352's ItemReader includes Spring Batch's ItemStream capabilities
ItemWriter / ItemStream	<b>ItemWriter</b>	JSR-352's ItemReader includes Spring Batch's ItemStream capabilities
JobExecutionListener	<b>JobListener</b>	
StepExecutionListener	<b>StepListener</b>	

## ■ Suitable for different applications

Spring Batch supports implementing reliable batch jobs with small effort. In few cases even no Java code is necessary.

It's very easy to customize existing parts at your needs. Different level of customization is possible:



# Questions and answers ...

Michael Beer

[michael.beer@trivadis.com](mailto:michael.beer@trivadis.com)

+41 58 459 51 90

Raffael Schmid

[raffael.schmid@trivadis.com](mailto:raffael.schmid@trivadis.com)

+41 58 459 52 34



BASEL BERN BRUGG LAUSANNE ZUERICH DUESSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA