

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



About us



Michael Beer Senior Consultant



Raffael Schmid
Consultant

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

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



AGENDA

Initial position

Requirements and Topics

Lessons learned

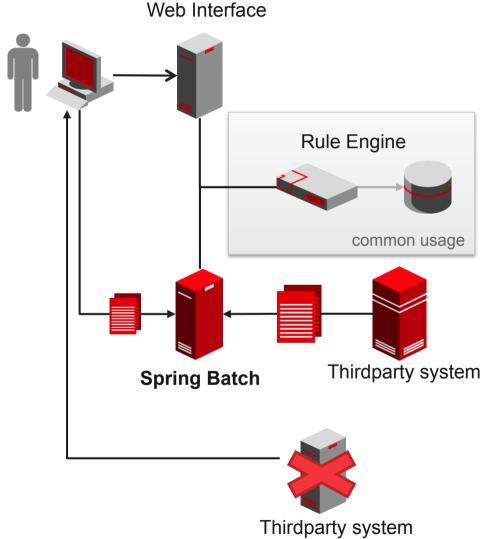
Conclusion



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

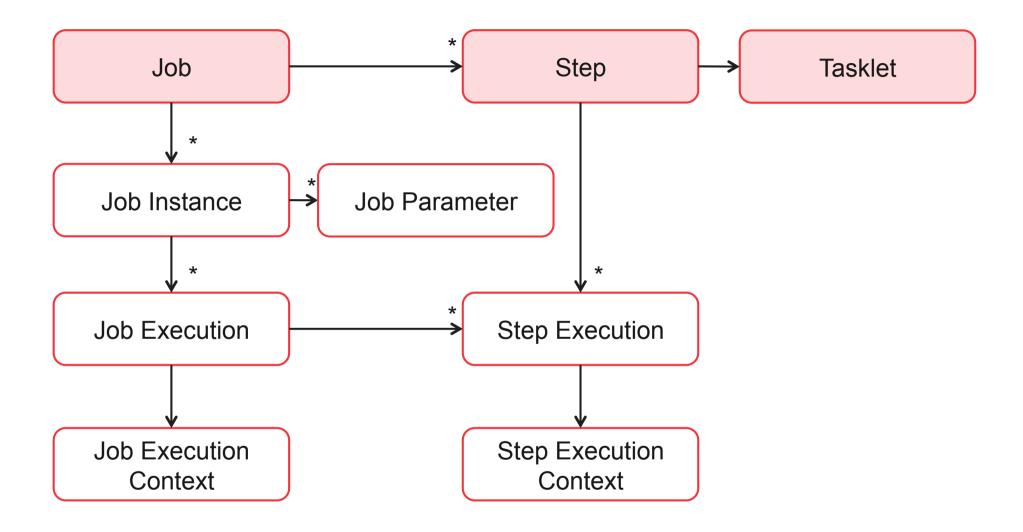


Why we used Spring Batch?

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



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

- parallelized on multiple threads
- order of execution matters (in case of partitioning)
- 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

- we do not stop job in case of an error
- 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

- 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

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



Requirement: Trigger jobs

Performance

Reprocess failed items

Summary mail

Gather detailed job information

Trigger jobs

Control load

Deactivate jobs

- 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

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



Requirement: Deactivate jobs

Performance

Reprocess failed items

Summary mail

Gather detailed job information

Trigger jobs

Control load

Deactivate jobs

- 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 Handling Monitoring & Tracing Monitoring & Tracing Monitoring & Tracing Sourcing

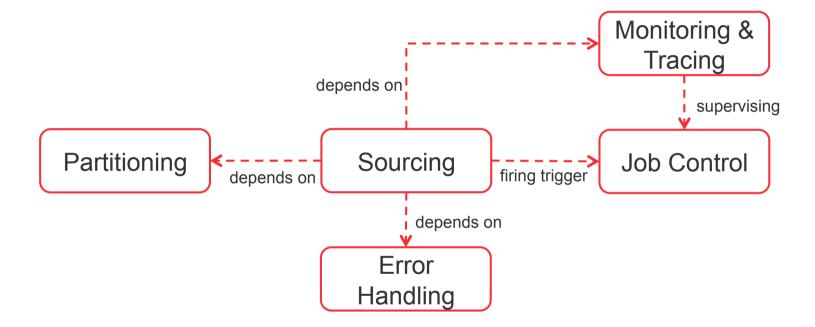


Lessons learned

Initial Requirements Lessons learned Conclusion

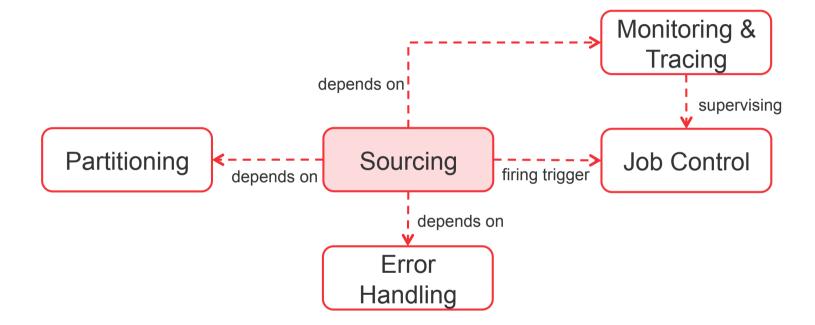


Topics we will cover





Sourcing





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	Н	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



Sourcing

Allows partitioning, single record execution

PARTITION	LINE	TYPE	STATE	FIELD01	FIELD02	FIELD03	FIELD04
PT_000123	1	Н	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;
```

Reexcution of failed records

```
SELECT * FROM LOAD

WHERE PARTITION = 'PT_000123'

AND STATE = '01'
```

Execution of single record

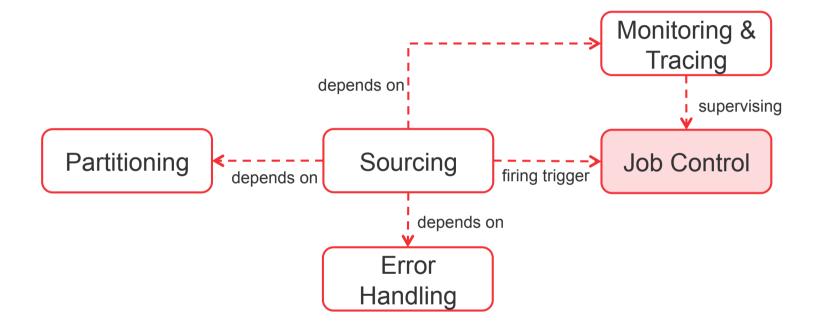
```
SELECT * FROM LOAD

WHERE PARTITION = 'PT_000123'

AND LINE = 8
```



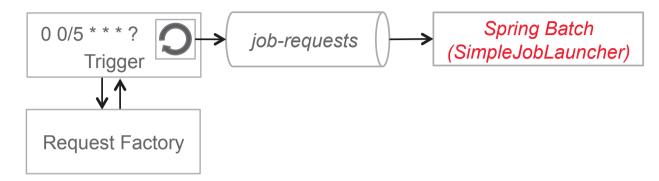
Job Control





Job Control

Simplest way of triggering a job



Define polling channel adapter (CronTrigger)

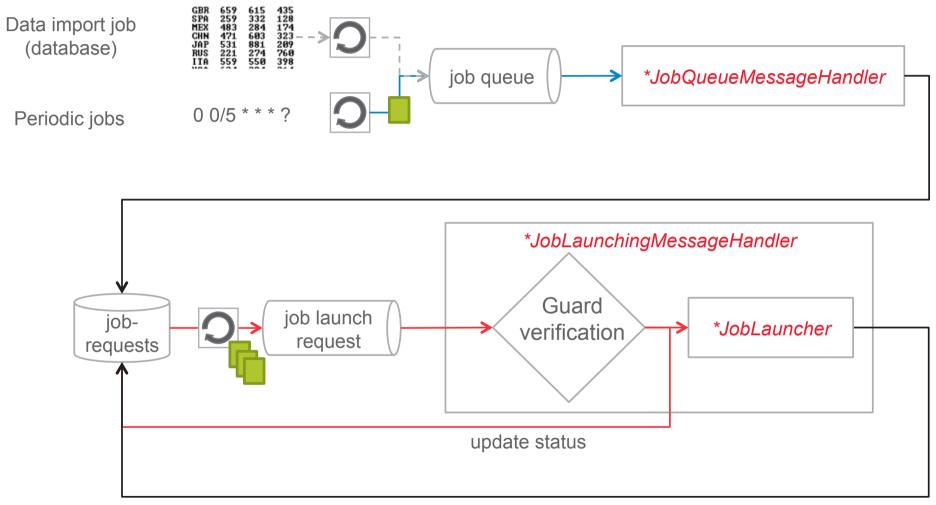
Or even simpler (PeriodicTrigger)



June 24th. 2014

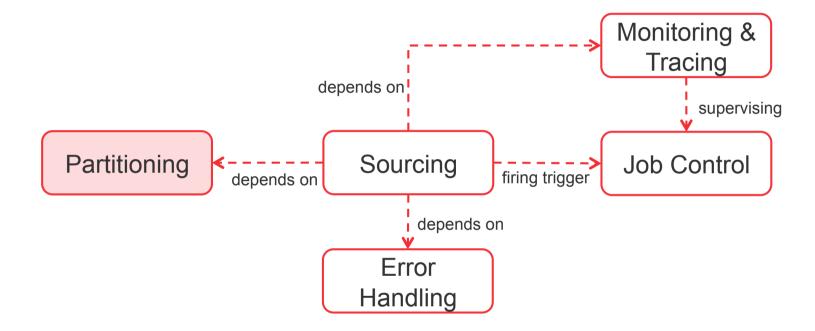
Job Control

Persist job launch requests into database









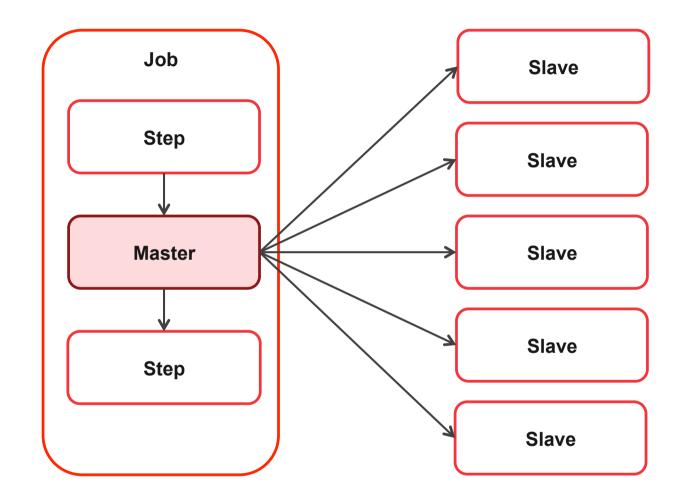


Performance

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

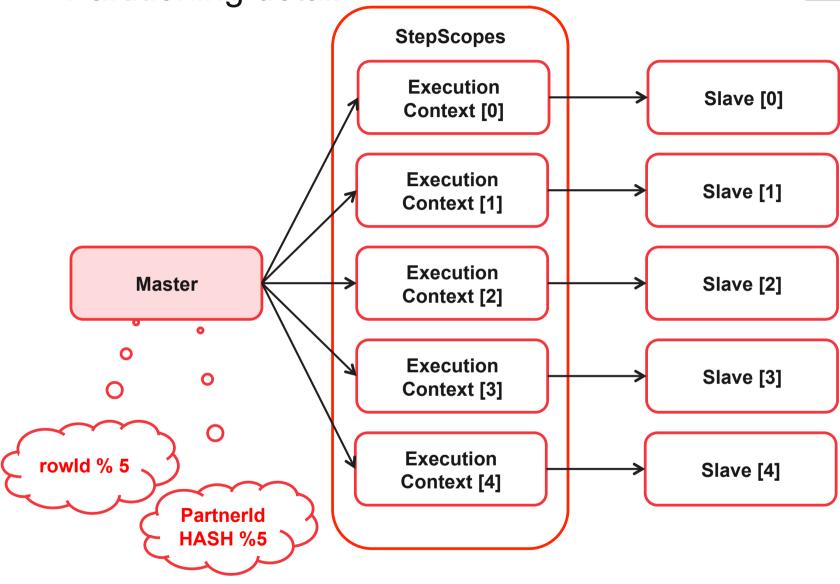


Partitioning overview





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	Insert	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	Update	04.12.2013 03.02.2013





2014 © Trivadis

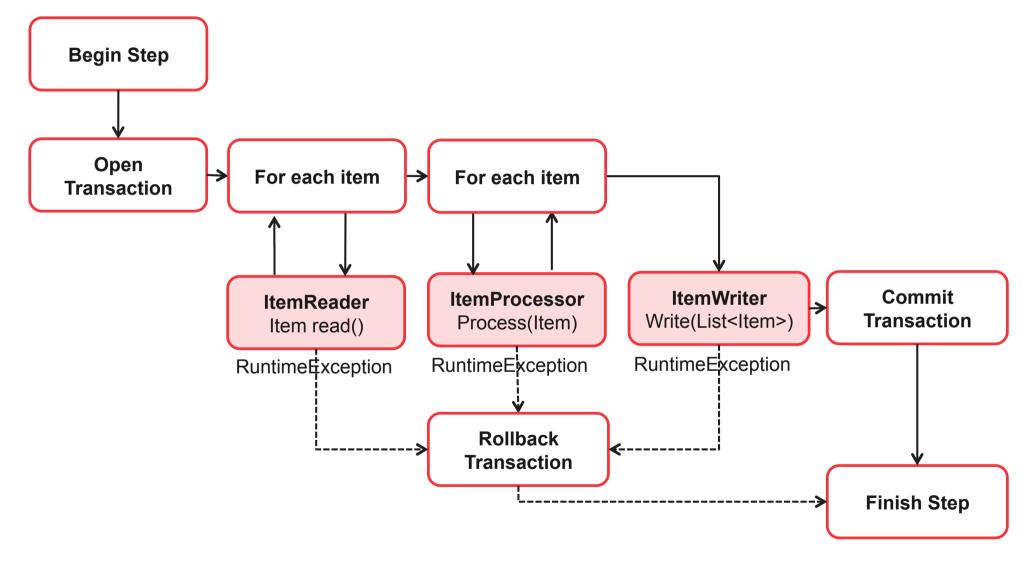
■ Partitioning detail – Spring Batch Admin

Property	Value
ID	0
Job Name	csv-partition-sample-job
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	COMPLETED
partitionSlave:partition3	4	4	3	0	00:00:00	COMPLETED
partitionSlave:partition2	4	4	3	0	00:00:00	COMPLETED
partitionSlave:partition4	4	4	3	0	00:00:00	COMPLETED
partitionSlave:partition1	4	4	3	0	00:00:00	COMPLETED
partitionSlave:partition0	4	4	3	0	00:00:00	COMPLETED



Performance – Reader / Processor / Writer





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

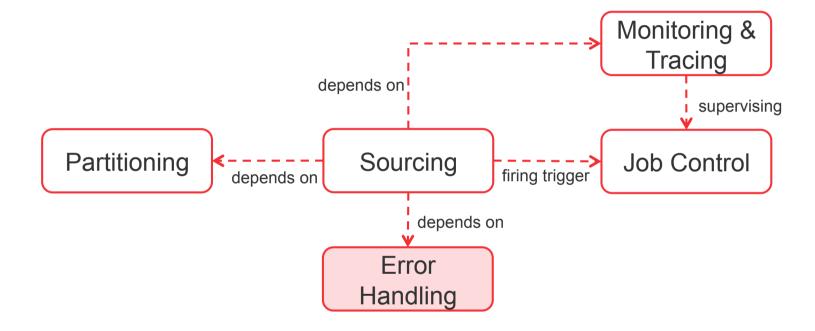
Error on each item

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



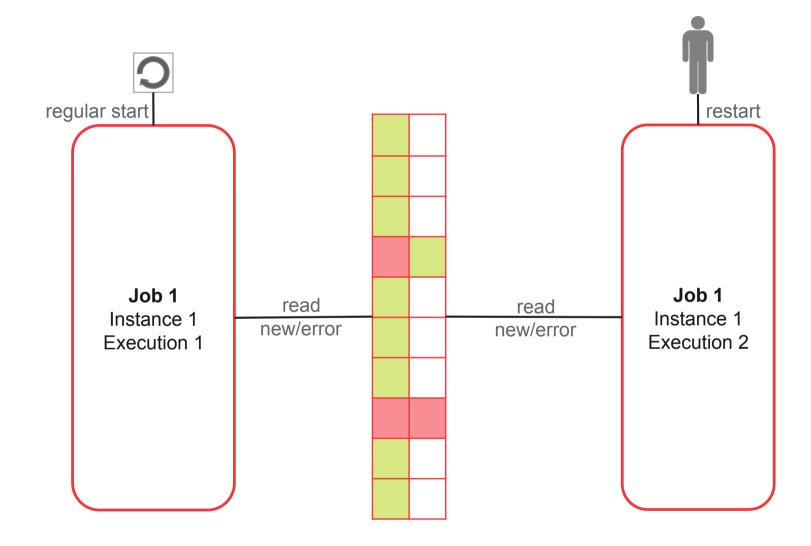
Error Handling





Restartability

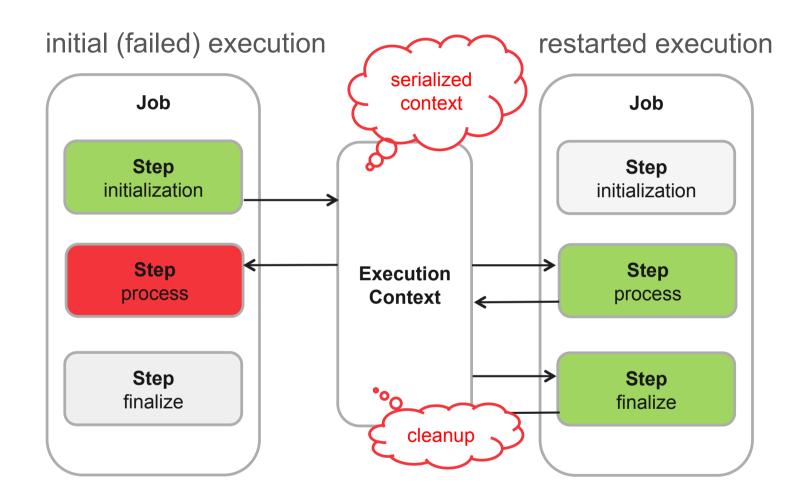






Restartability configuration







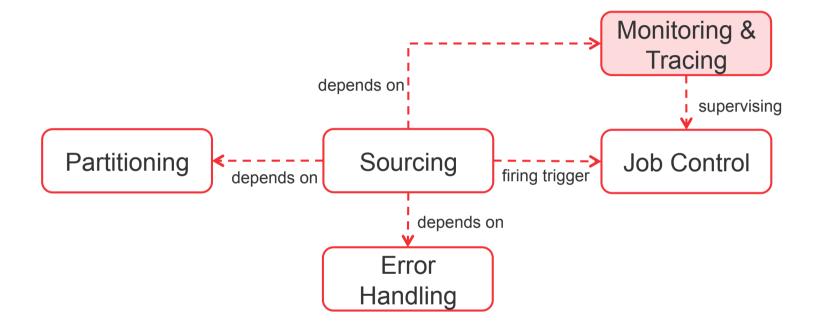
Item based error handling



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

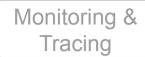


Monitoring





Monitoring – Spring Batch Admin



Spring Batch Admin



Exit Message

org.springframework.batch.item.ltemStreamException: 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

 $org. spring framework. a op. support. Delegating Introduction Interceptor. do Proceed (Delegating Introduction Interceptor. java: 131)\ at$

Caused by: <u>java.lang.IllegalStateException</u>: 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.slimplestephandler.nandlestep(slimplestephandler.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

 $org. spring framework. batch. core. launch. support. Simple Job Launcher \$1. run (Simple Job Launcher. java: 120)\ at$

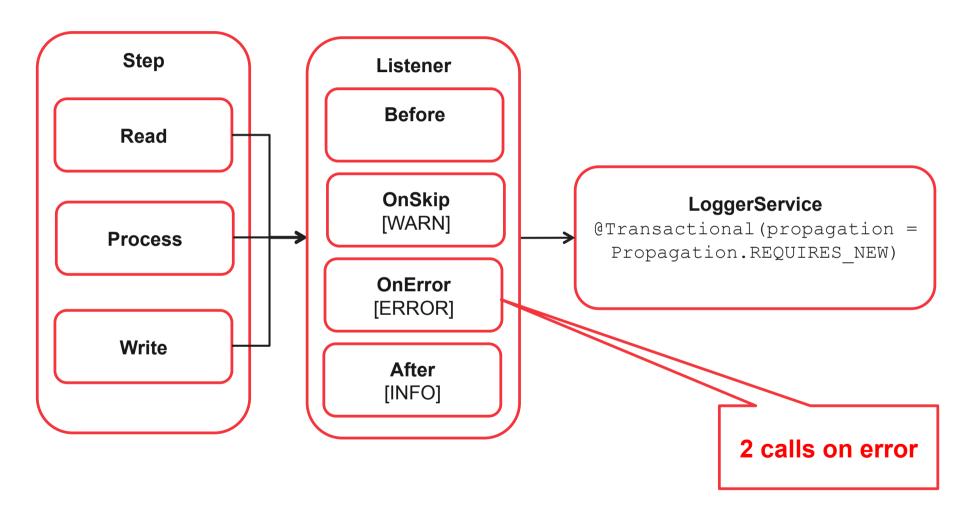
Java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145) at Java.util.concurrent.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:615)

@ Copyright 2009-2010 SpringSource. All Rights Reserved.

Contact SpringSource

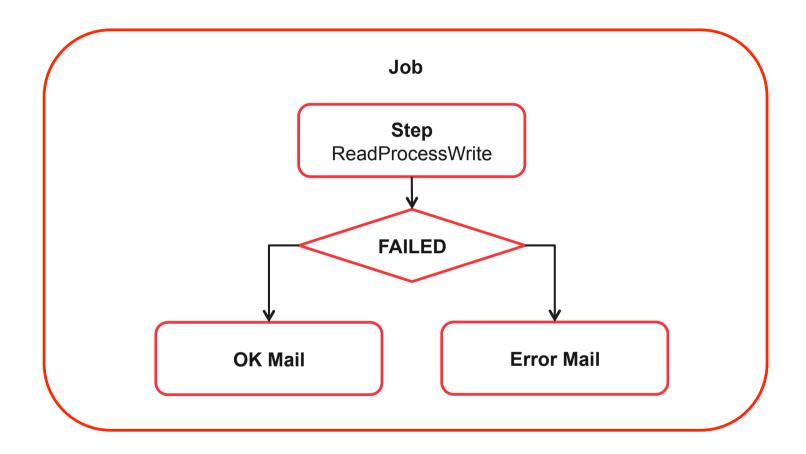


Monitoring – logging to a database



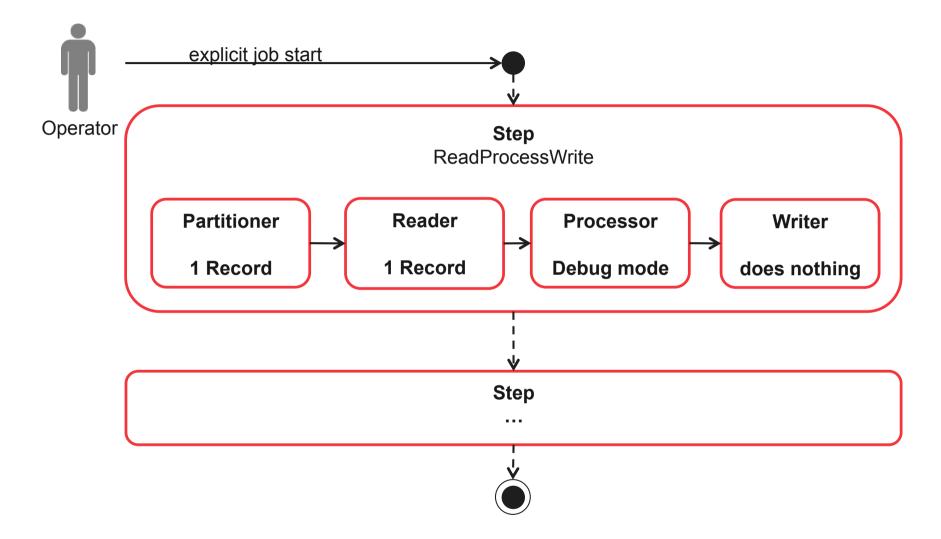


Monitoring - Mail



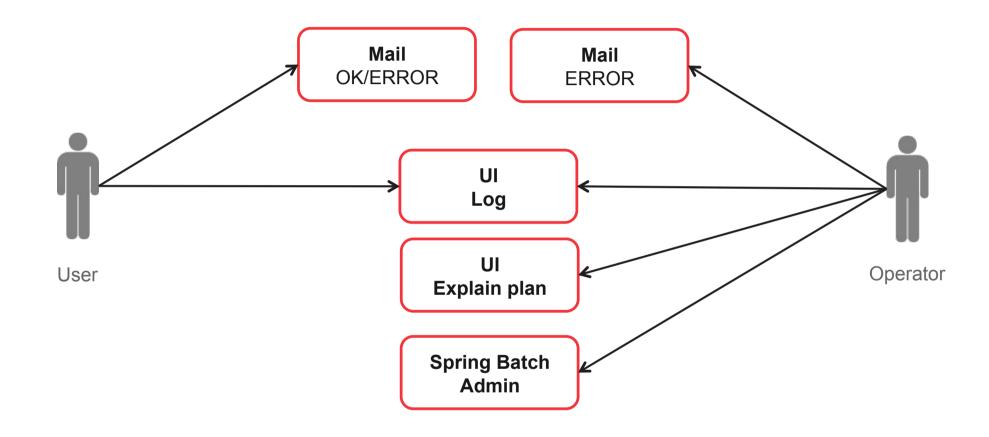


Tracing – Explain plan





Monitoring / Tracing summary





Conclusion

Initial position

Requirements and Topics

Lessons learned

Conclusion



2014 © Trivadis

Things to think about – Framework evaluation

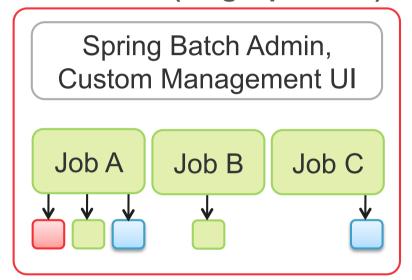
- Testability
 - Spring Batch provides lots of helpers to write JUnit tests
- Monitoring
 - Spring Batch Admin
- Scalability
- Extensibility
 - Easy to write extentions (Job Launch, Guards)
- Out of the box components

All 155 SE tests in the JSR-352 TCK passed (Spring Batch 3.0)



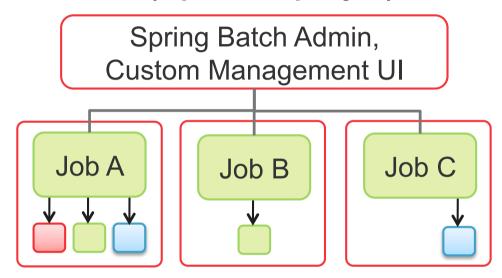
■ Things to think about – Monolitic vs. Modular

Monolithic (single process)



 Integrate batch jobs into web application (use Spring Batch Admin)

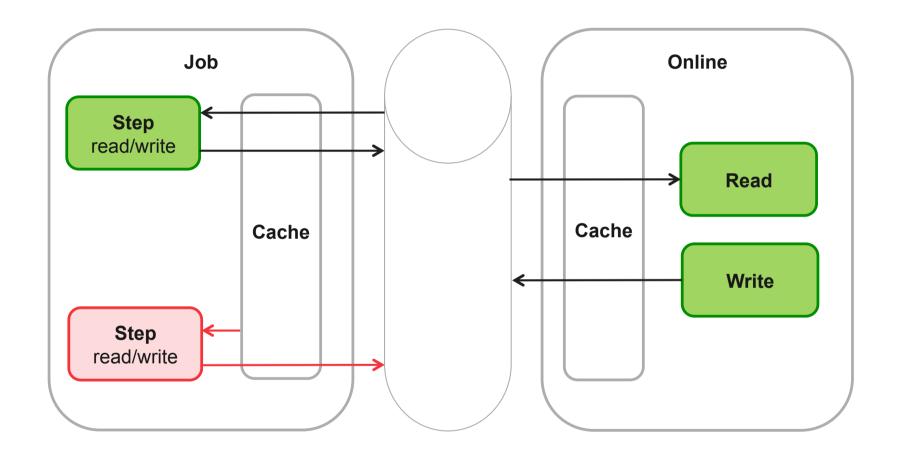
Modular (1 process per job)



- Start Jobs individually on command line (use external scheduler)
- Use Spring XD



■ Things to think about – Caching







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

