WELCOME

Spring Batch - Field Report

Michael Beer,
Raffael Schmid

September 27th, 2013

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

trivadis
makes IT easier.

AGENDA

1. **Spring Batch framework and lessons learned**

2. Execution environment – Field Report

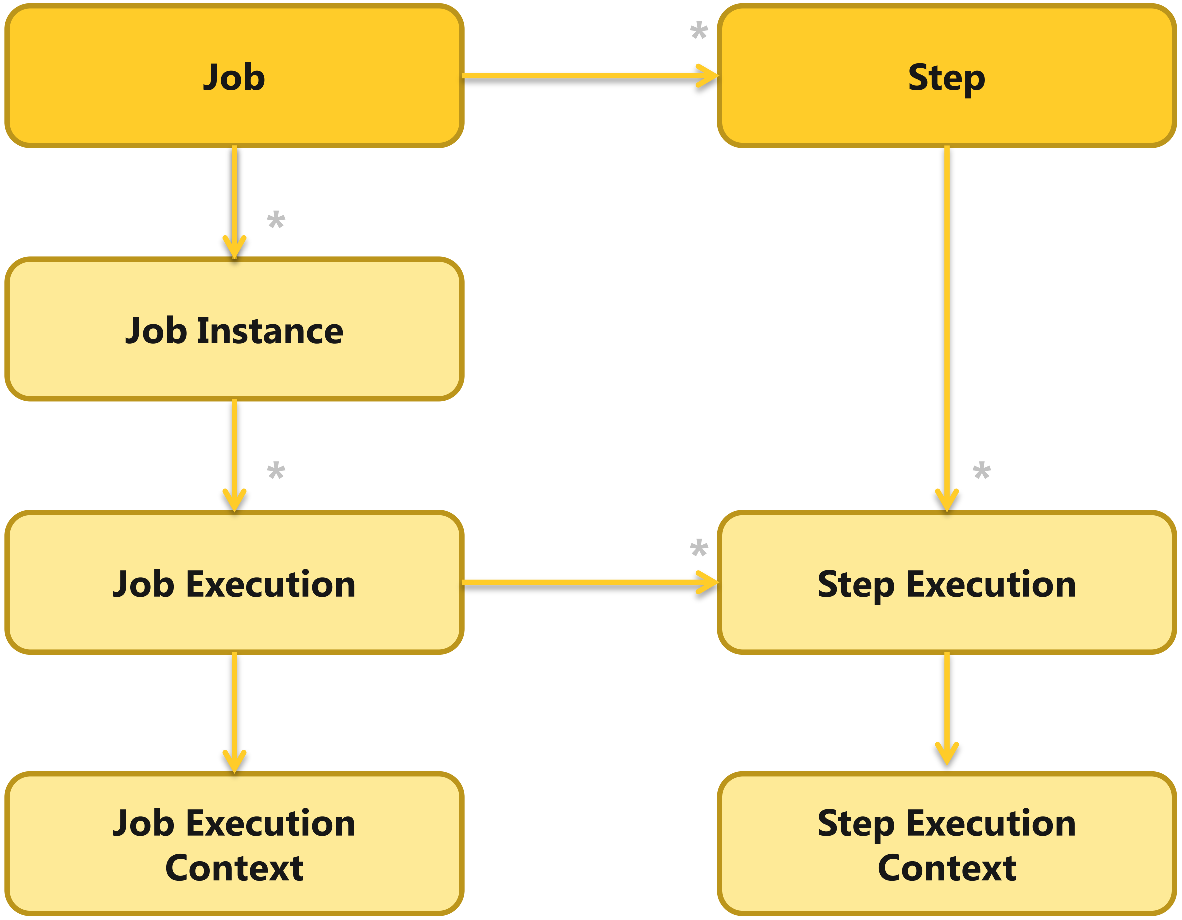
3. Batch Applications for the Java Platform (JSR-352)

trivadis
makes IT easier.

Roadmap

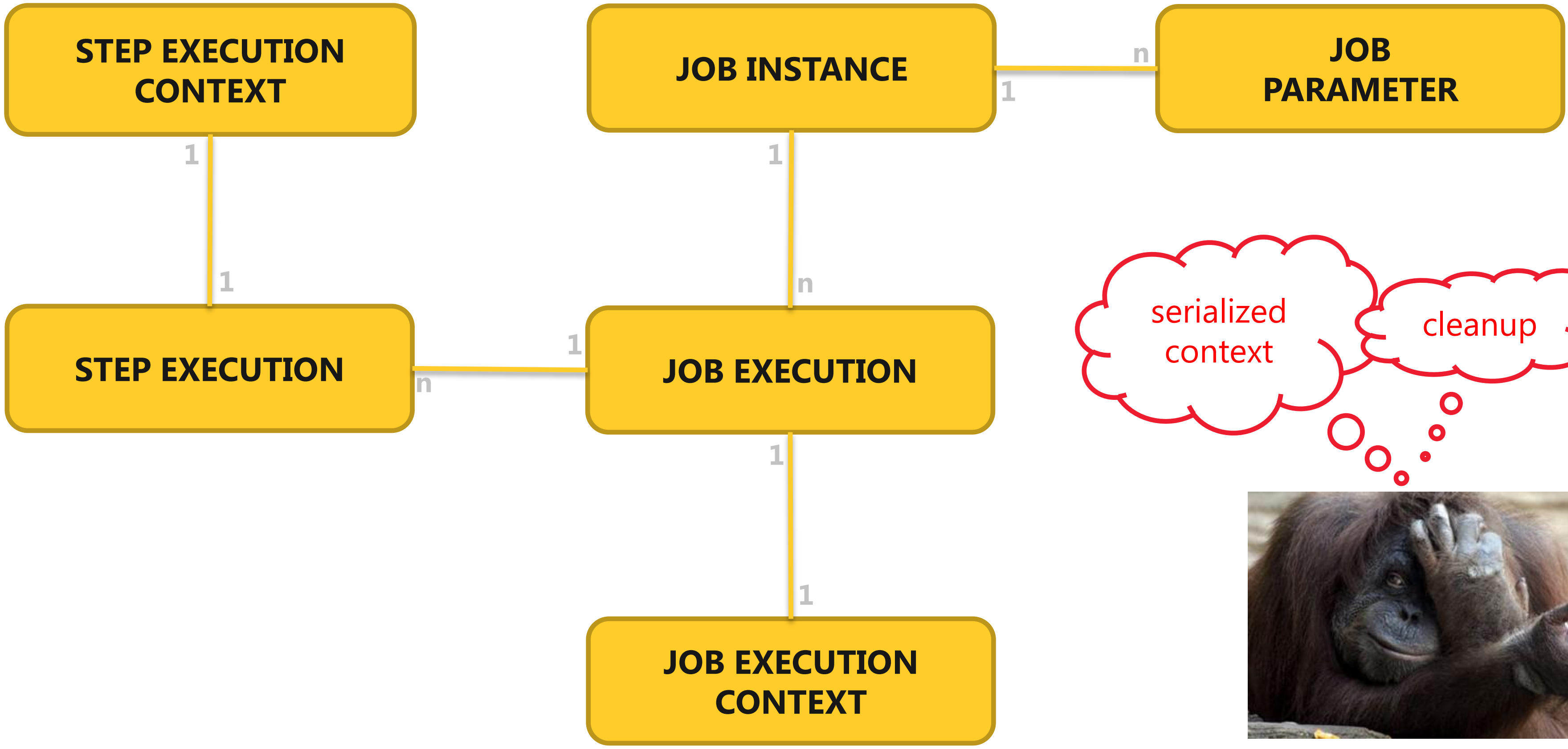
Version	Description
1.x	Single-process, possibility multi-threaded execution
2.x	New features enabling an application to scale to multiple processes
Java 7	**JSR-352 - Batch Applications for the Java Platform**
3.0 M1	Alignement with JSR-352 (70/155 TCK)

Job/Step Stereotypes



trivadis makes IT easier.

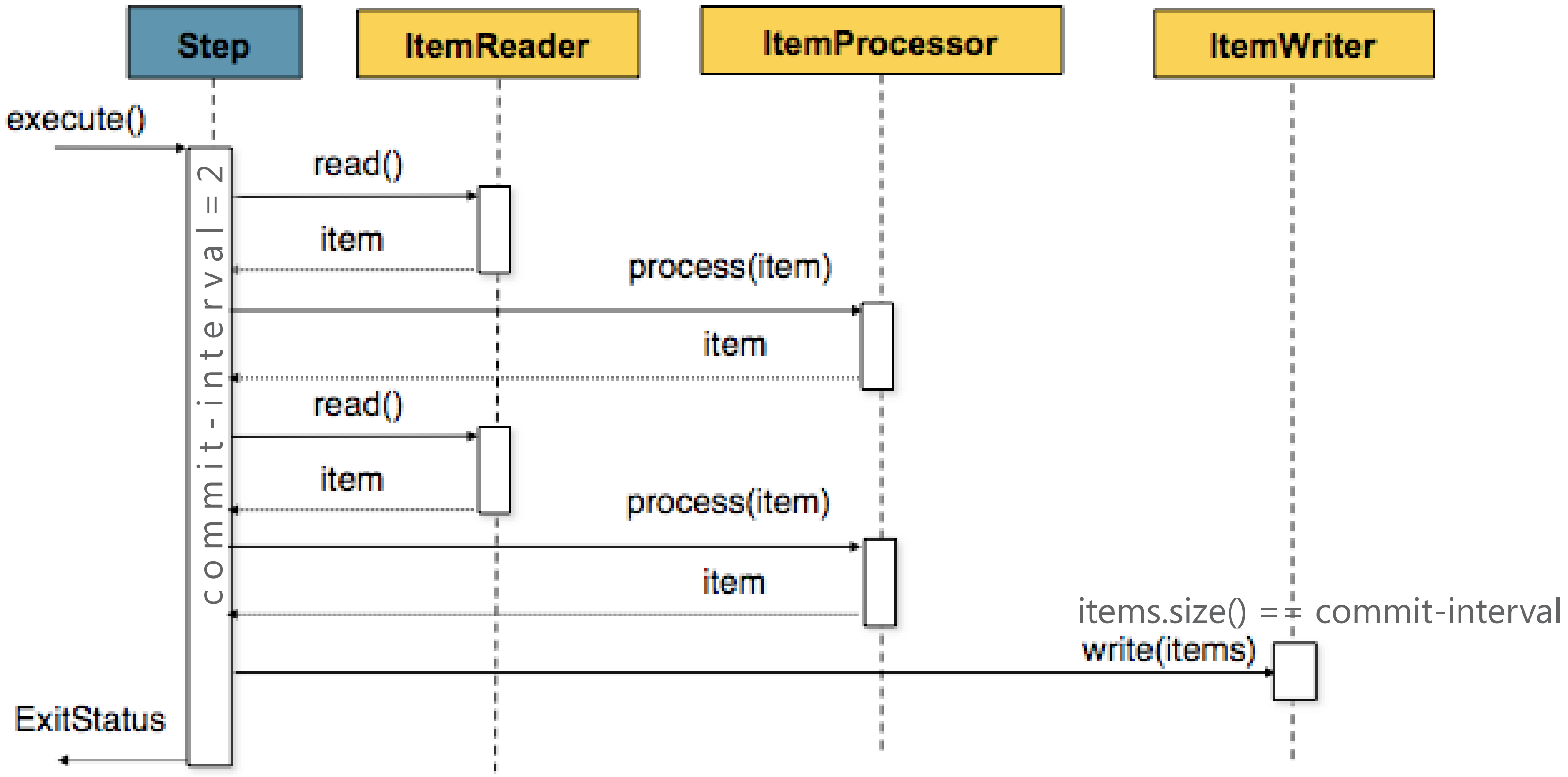
Meta-Data Schema



Job/Step/Tasklet configuration

```
<job id="job" restartable="false">
     <validator ref="validator" />
     <step id="step">
         <partition step="partitionerStep" partitioner="partitioner" />
     </step>
     <decision id="decision" decider="decider">
         <next on="*" to="nextStep" />
         <end on="FAILED" />
     </decision>
</job>

<step id="partitionerStep">
     <tasklet transaction-manager="transactionManager">
         <chunk reader="pagingItemreader" processor="processor"
                writer="writer" commit-interval="10"
                skip-policy="skipPolicy"/>
         <listeners>
             <listener ref="listener" />
         </listeners>
     </tasklet>
</step>
```

Chunk / Read / Process / Write



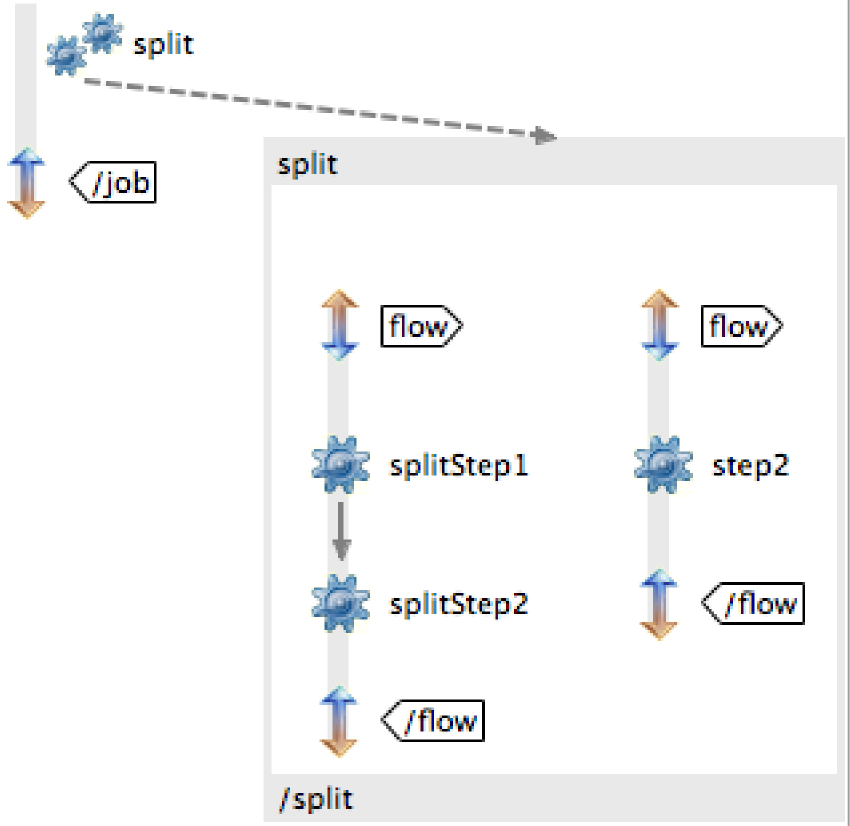
(Source: http://docs.spring.io/spring-batch/reference/html/configureStep.html)

Scalability

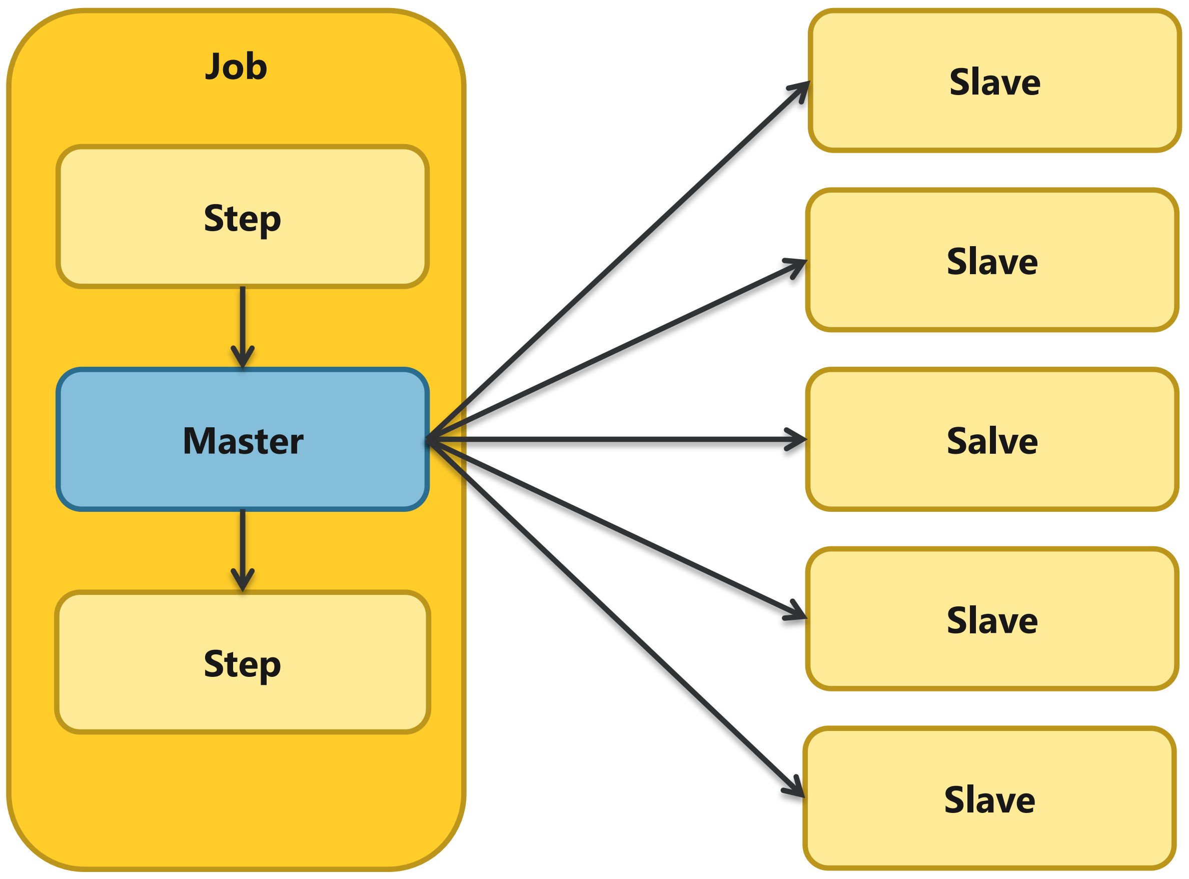
Typ	Local/Remote	Description
Multi-threaded Step	Local	A step is multithreaded (TaskExecutor)
Parallel Steps	Local	Executes steps in parallel using multithreading
Partitioning Step	Local Remote	Partitions data and splits up processing
Remote Chunking	Remote	Distributed chung processing to remote nodes

Parallel Steps

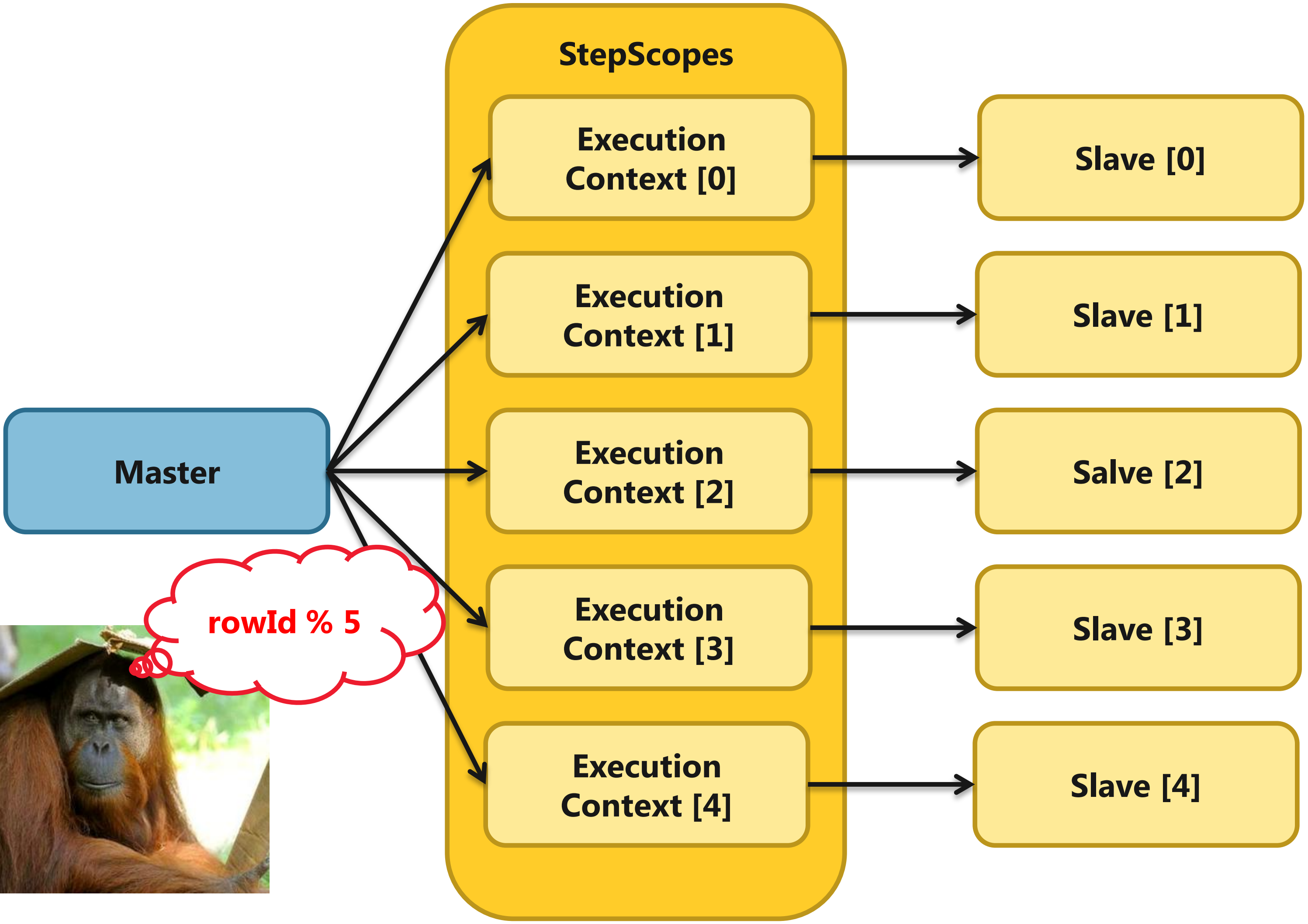
```
<split id="split“>
      <flow>
         <step id="splitStep1“ next="splitStep2"/>
         <step id="splitStep2"/>
      </flow>
      <flow>
         <step id=“step"/>
      </flow>
</split>
```

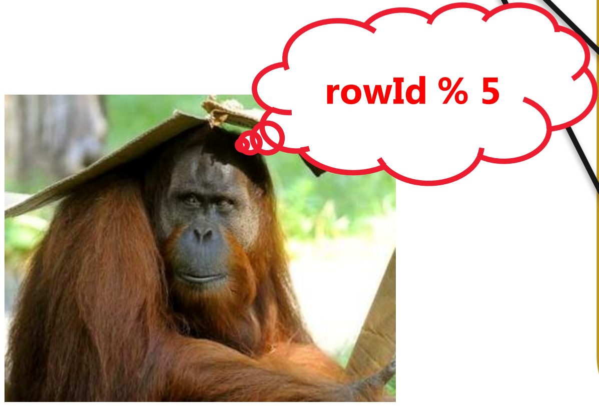


Partitioning overview

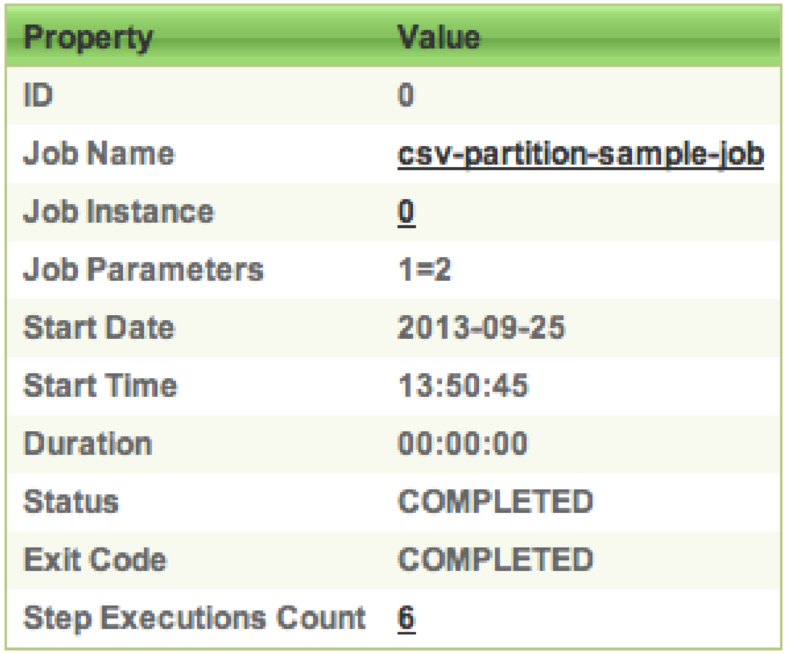


Partitioning detail

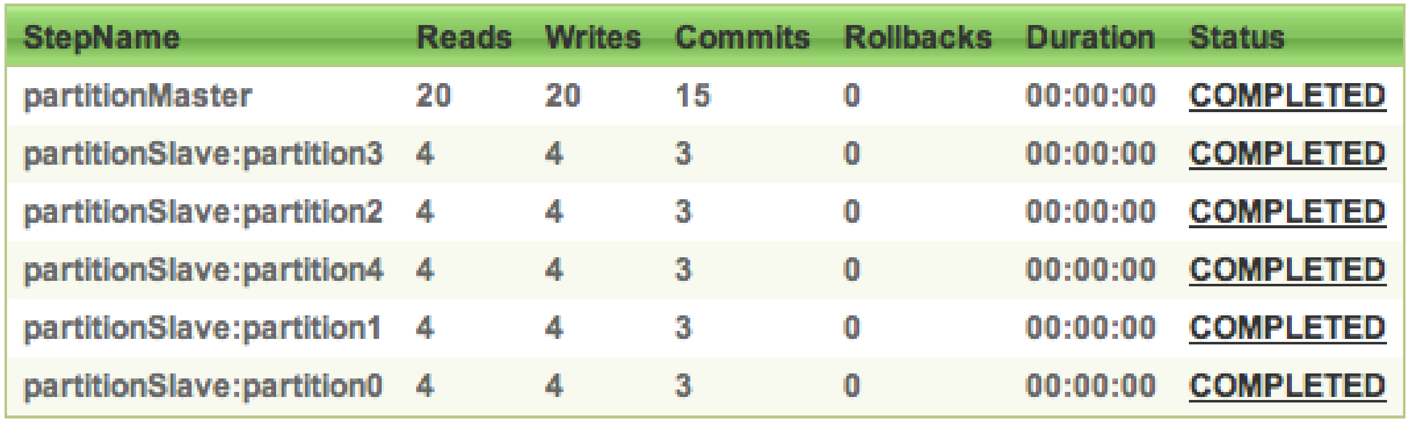




Partitioning detail – Spring Batch Admin

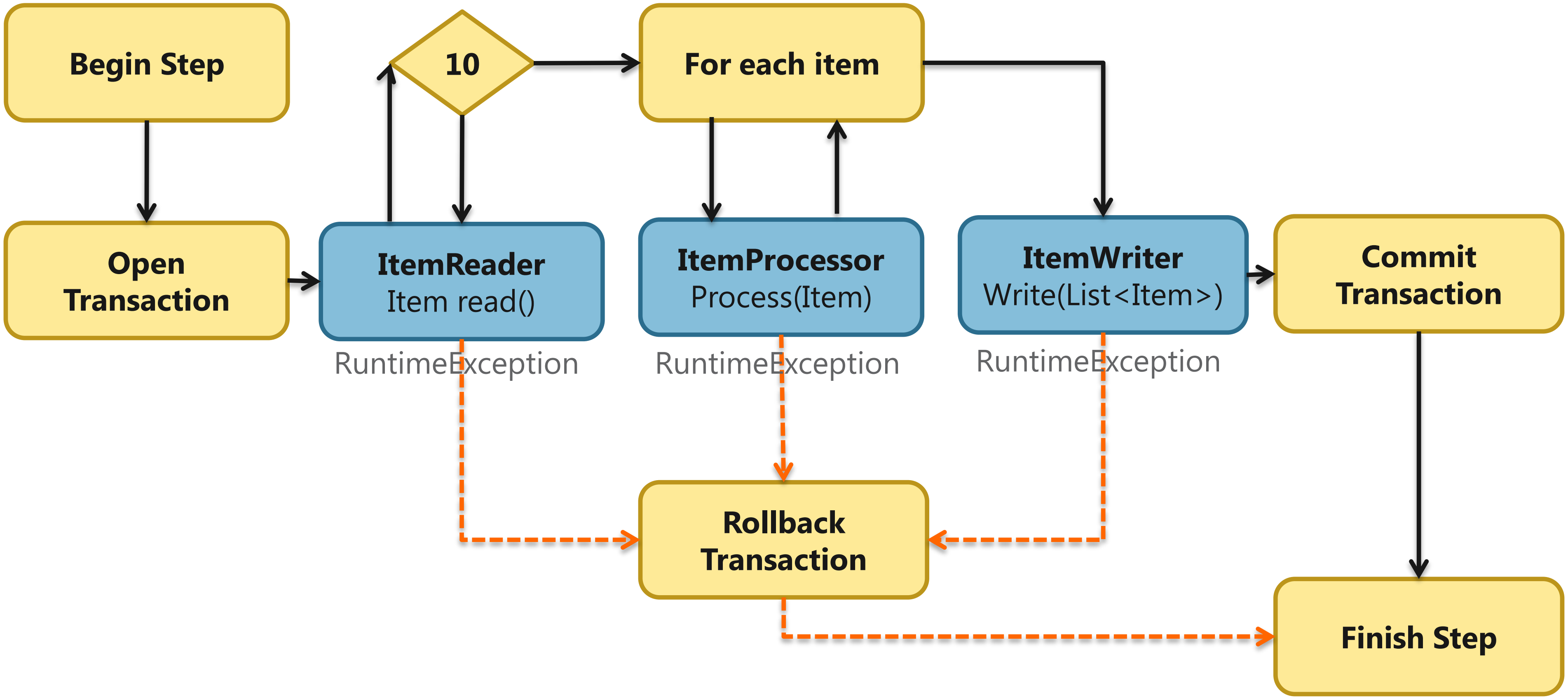


Property	Value
ID	0
Job Name	csv-partition-sample-job
Job Instance	0
Job Parameters	1=2
Start Date	2013-09-25
Start Time	13:50:45
Duration	00:00:00
Status	COMPLETED
Exit Code	COMPLETED
Step Executions Count	6



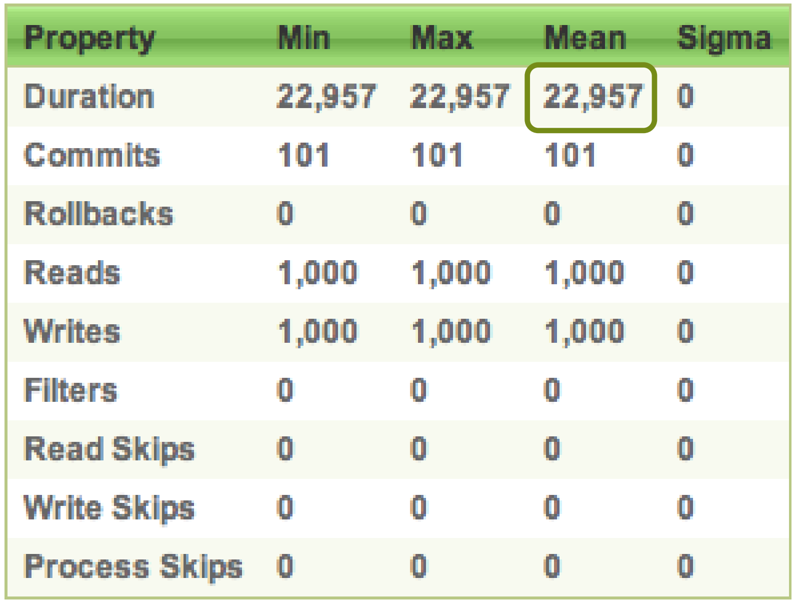
StepName	Reads	Writes	Commits	Rollbacks	Duration	Status
partitionMaster	20	20	15	0	00:00:00	COMPLETED
partitionSlave:partition3	4	4	3	0	00:00:00	COMPLETED
partitionSlave:partition2	4	4	3	0	00:00:00	COMPLETED
partitionSlave:partition4	4	4	3	0	00:00:00	COMPLETED
partitionSlave:partition1	4	4	3	0	00:00:00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

Transaction



Performance skip all / chunk processing (2/2)

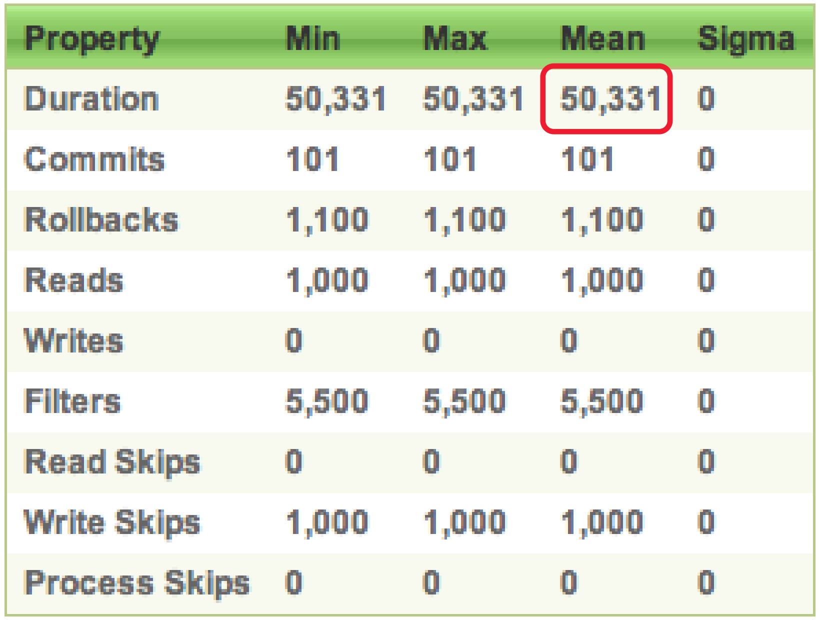
No error



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

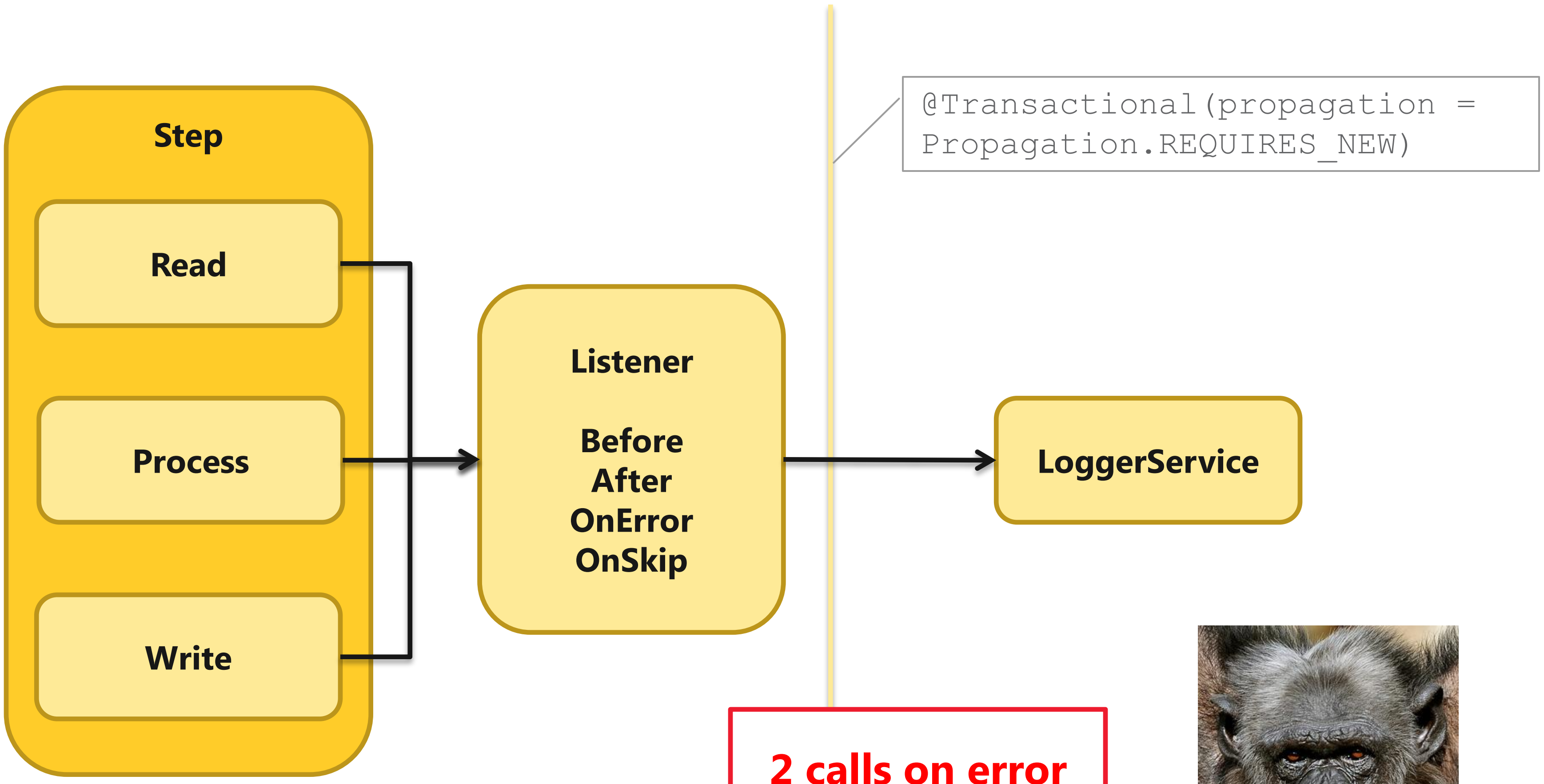
Error on each item



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

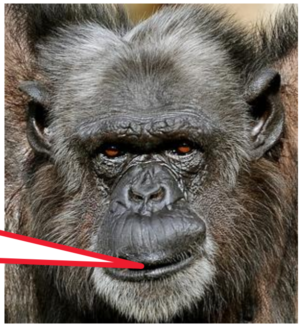
- ~ 50 sec
- 1‘100 Rollbacks
- 5‘500 Filter
- 1‘000 Write Skips

Listener



```
@Transactional(propagation =
Propagation.REQUIRES_NEW)
```

2 calls on error in writer



Skip/Retry/Restart – Bulletproof Job

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
Restart	After an excution failure	Restarts a job instance where the last execution failed	On job launch

Test

- End-To-End Testing of Batch Jobs
- Testing Individual Steps
- Testing Step-Scoped Components
- Validating Output Files
- MetaDataInstanceFactory
  - JobExecution
  - JobInstance
  - StepExecution

General Principles and Guidelines for Batch Architectures

- Simplify and avoid building complex logical structures
- Think about the overhead when using ORM
  - Caching
  - Lazy loading
- Carefully design application I/O
  - Read, write once
  - Use cursors
- Always assume the worst with regard to data integrity
- Plan and execute stress tests as early as possible

AGENDA

1. Spring Batch framework and lessons learned

2. **Execution environment – Field Report**

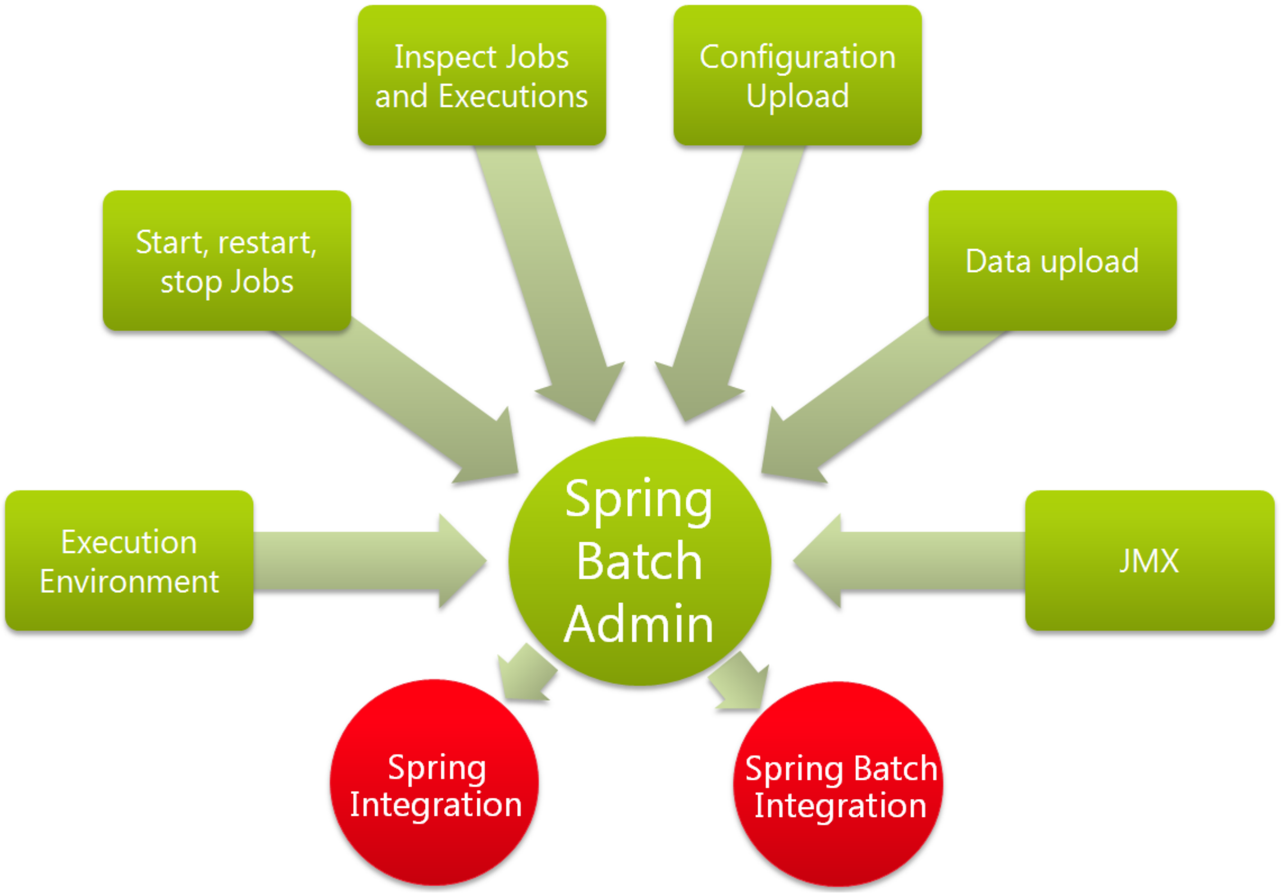
3. Batch Applications for the Java Platform (JSR-352)

Requirements to the execution environment

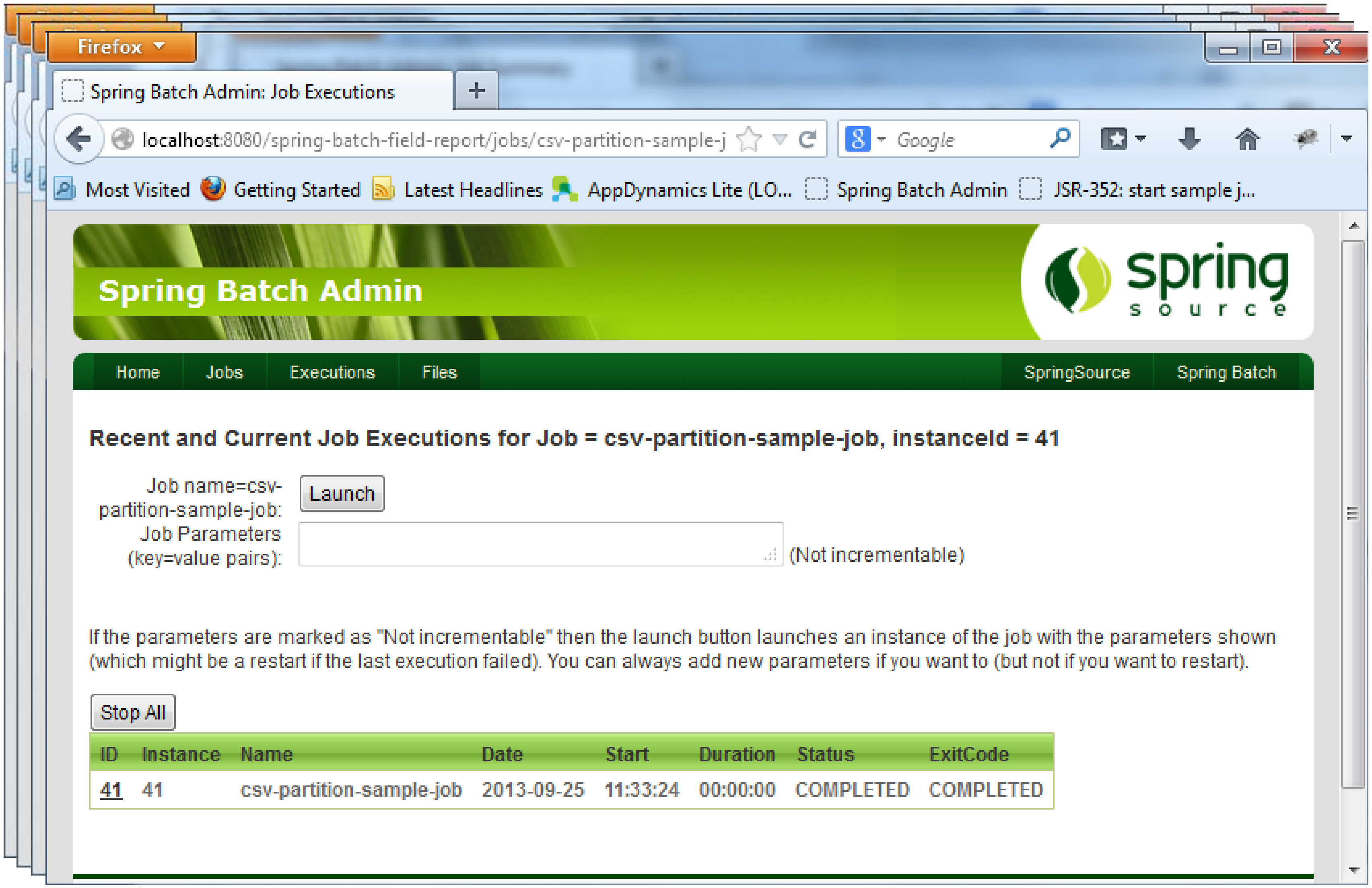
1. **Administrative User Interface**
   „Jobs can be started, stopped, monitored over a Web Interface."
2. **Trigger jobs periodically or out of database events**
   „Jobs must be triggered either by cron expression, fixed rate, fixed delay or due to new data in the staging area."
3. **Control execution of jobs**
   „Postpone executions due to inter-job dependencies or control the database-load."
4. **Detailed execution log**
   „A detailed execution log needs to be available over a web interface."
5. **Gather diagnostic information**
   „Only to know that an error happened might not be sufficient. Sometimes further diagnostic information is necessary."

trivadis
makes IT easier.

Administrative User Interface



Administrative User Interface



Recent and Current Job Executions for Job = csv-partition-sample-job, instanceId = 41

Job name=csv-partition-sample-job: [Launch]

Job Parameters (key=value pairs): (Not incrementable)

If the parameters are marked as "Not incrementable" then the launch button launches an instance of the job with the parameters shown (which might be a restart if the last execution failed). You can always add new parameters if you want to (but not if you want to restart).

[Stop All]

ID	Instance	Name	Date	Start	Duration	Status	ExitCode
41	41	csv-partition-sample-job	2013-09-25	11:33:24	00:00:00	COMPLETED	COMPLETED

Spring Batch Admin - Setup

Most probably the toughest job for a Spring application!

1. Configure the library dependencies
2. Setup or enrich root context

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath*:/.../webapp-config.xml</param-value>
</context-param>
```

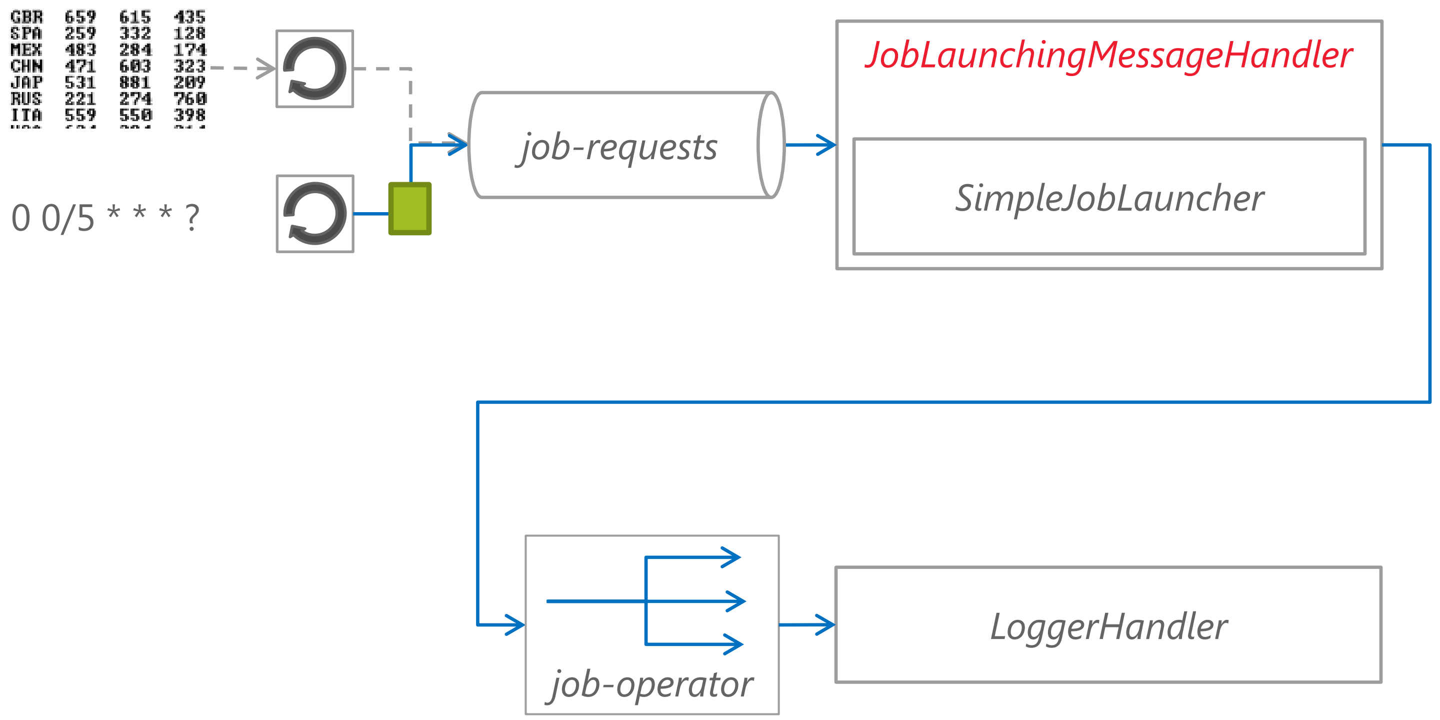
3. Configure servlet and mapping

```
<servlet>
  <servlet-name>Batch Servlet</servlet-name>
  <servlet-class>org.sfw.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath*:/.../servlet-config.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

Trigger jobs periodically or out of database events

Data import job (database)

Periodic jobs



Trigger jobs periodically or out of database events

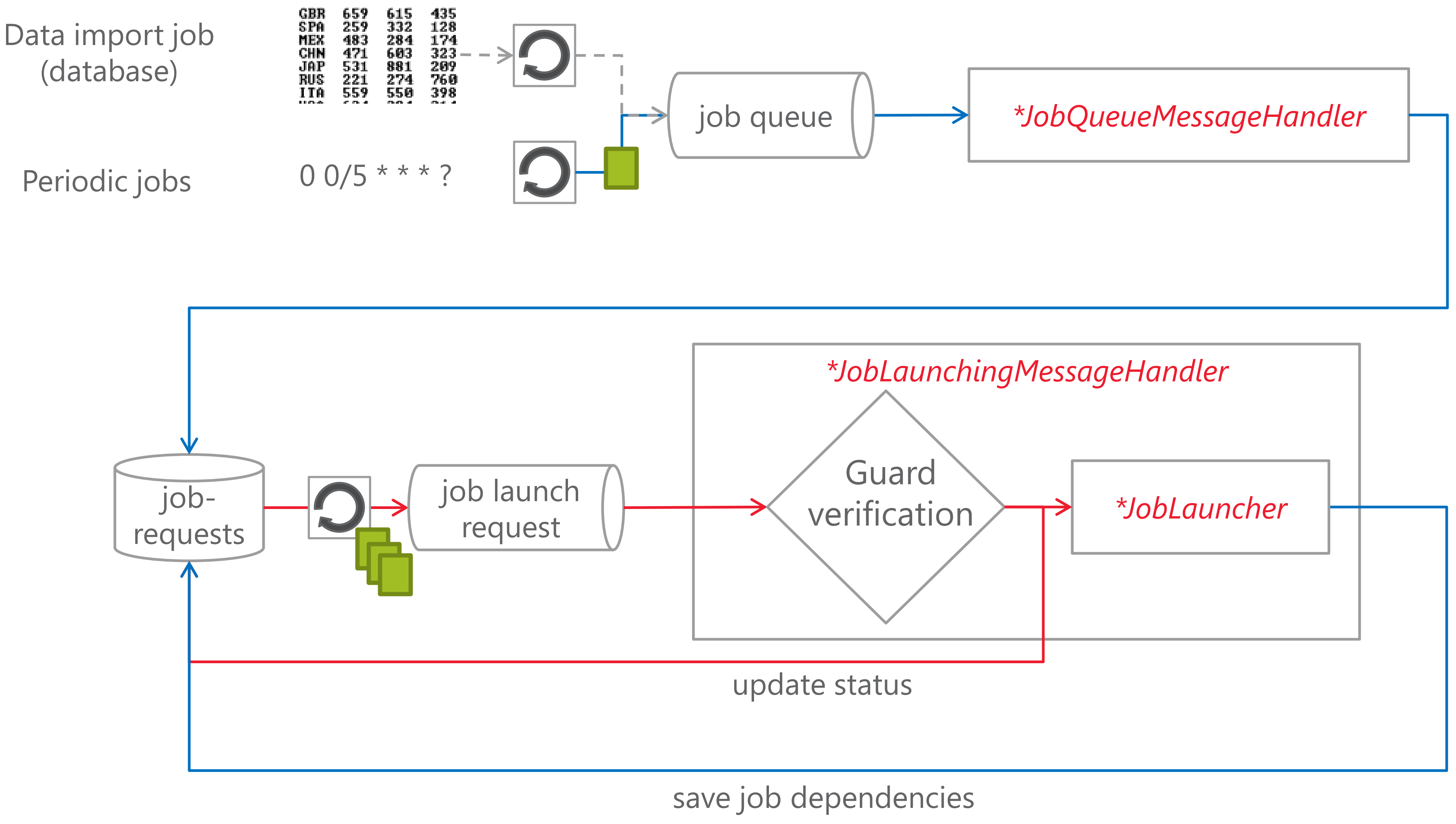
```
<bean id="factory" class="job.LoadJobLaunchRequestFactory">
  <constructor-arg ref="jobRegistry" />
</bean>

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

define polling channel adapter

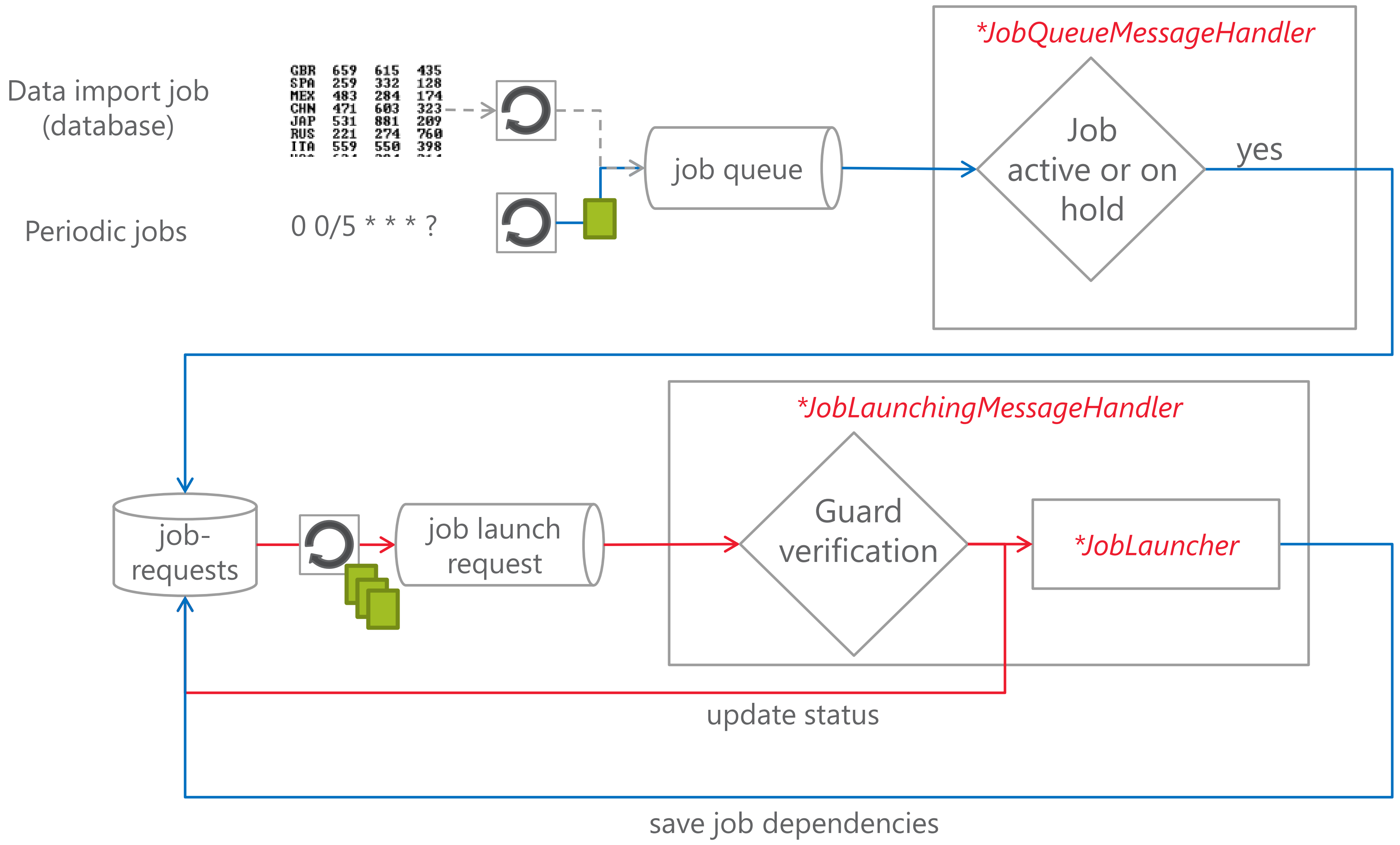
```
public class LoadJobLaunchRequestFactory {

  ...
  public JobLaunchRequest create() throws NoSuchJobException {

    JobParameters jobParams = new JobParametersBuilder()
      .addDate("random", new Date()).toJobParameters();
    return new JobLaunchRequest(jobLocator.getJob(NAME), jobParams);
  }
}
```

factory to create job launch request

Control execution time of job (e.g. postpone)



Disable job execution or set on hold



Detailed execution log

- All our jobs write an execution log, the collected information will be sent to a shared mailbox (evt. file submitter) at the end of a job
- Data will be written to a database, therefore a nested transaction is used this times (critical for success messages)
- *A message consists of the following fields:*
  - *Time*
  - *Thread name*
  - *Message*
  - *Status*
  - ***Reference (i.E. line number, object id, etc.)***
  - *Exception trace (opt)*

Gather diagnostic information (Explain Plan)

Problem:

Loading data involes calling a rule engine. The rule engine is able to collect diagnostic information (no default behaviour).

Solution 1: Store diagnostic information all the time

Solution 2: On error reprocess item in diagnostic mode

Solution 3: Let the user rerun the job for a single item in diagnostic mode

- *Involves adjusting partitioner and reader, the query might look like this*

```
select * from data_loader where partition_key = :partition_key
and record_number = nvl(:record_number, record_number)
```

- *Not applicable for file readers*

AGENDA

1. Spring Batch framework and lessons learned

2. Execution environment – Field Report

3. **Batch Applications for the Java Platform (JSR-352)**

JSR-352: Terminology

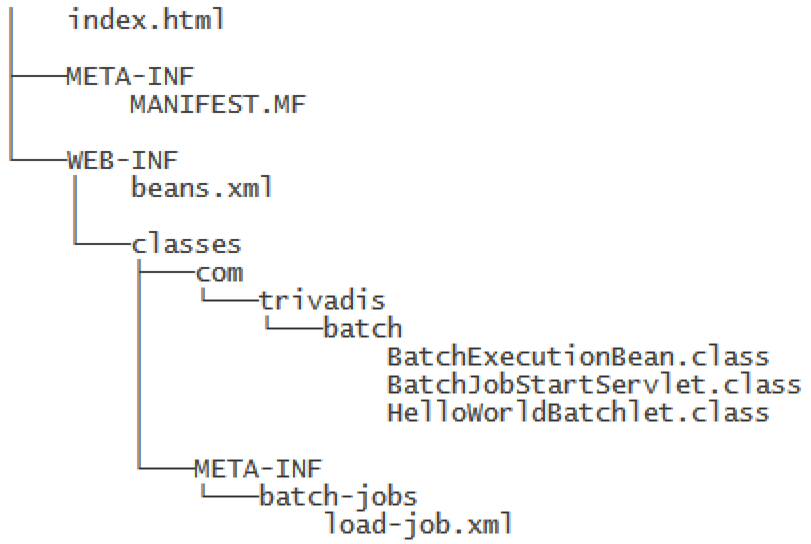
The terminology stays more or less the same: Job, Step, Chunk, Item, ItemProcessor, JobInstance, JobExecution.

The differences are summarized as follows:

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

JSR-352: Create and deploy a Batch Job

- Deployment as a Web Archive



```
│   index.html
│
├───META-INF
│       MANIFEST.MF
│
└───WEB-INF
    │   beans.xml
    │
    └───classes
        ├───com
        │   └───trivadis
        │       └───batch
        │               BatchExecutionBean.class
        │               BatchJobStartServlet.class
        │               HelloWorldBatchlet.class
        │
        └───META-INF
            └───batch-jobs
                    load-job.xml
```

- Deploy to JEE 7 compliant (e.g. Glassfish 4) application server

JSR-352: Create and deploy a Batch Job

```
public class HelloWorldBatchlet implements Batchlet {

  @Inject
  JobContext jobContext;

  @Inject
  StepContext stepContext;

  @Override
  public String process() throws Exception {
    ...

    return "SUCCESS";
  }

  @Override
  public void stop() throws Exception {

  }
}
```

JSR-352: Create and deploy a Batch Job

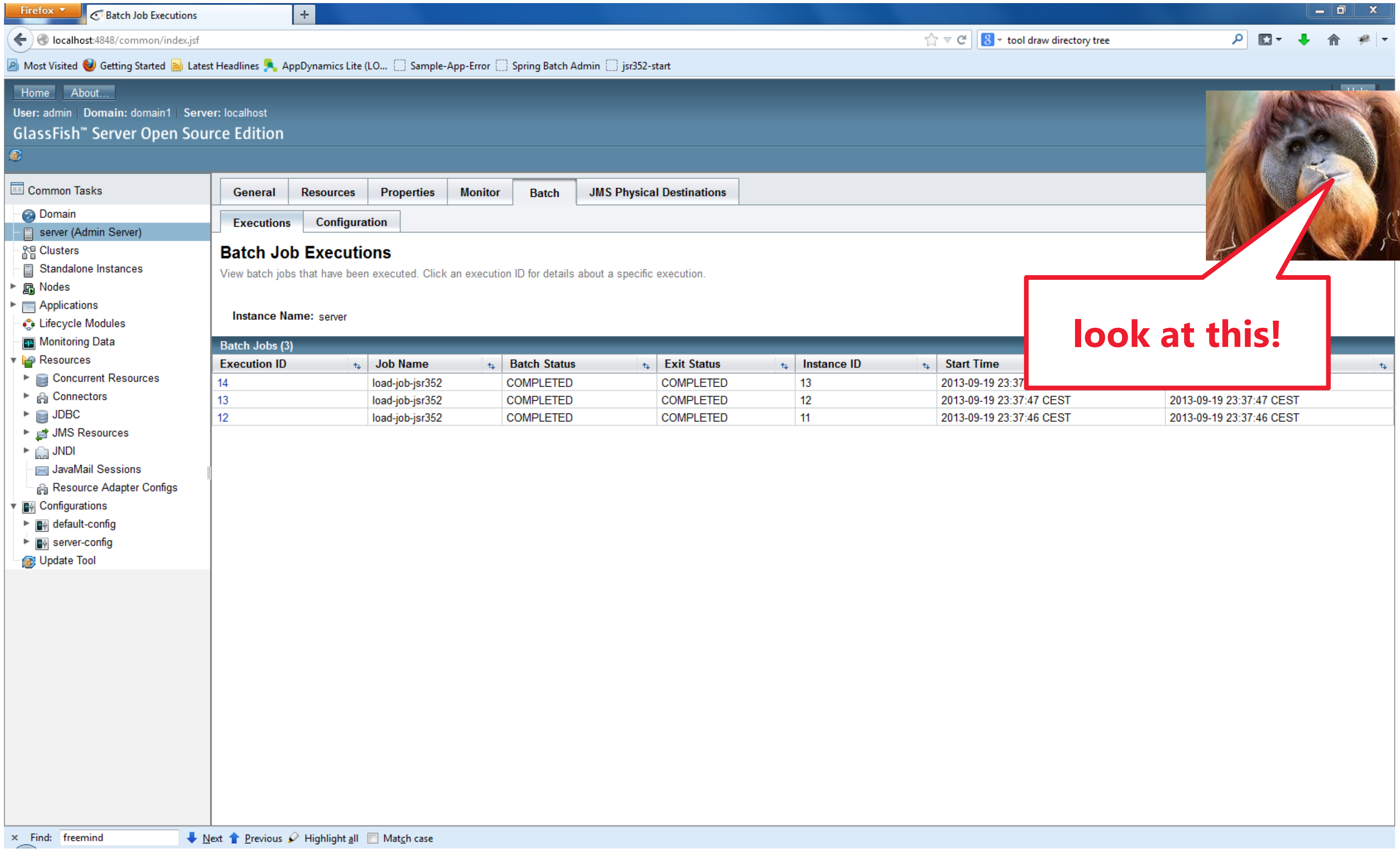
```
@Stateless
public class BatchExecutionBean {

  public long submitJob() {
    JobOperator operator = BatchRuntime.getJobOperator();
    Properties properties = new Properties();
    return operator.start("load-job-jsr352", properties);
  }
  ...
}
```

JSR-352: Create and deploy a Batch Job

```
<?xml version="1.0" encoding="UTF-8"?>
<job id="load-job" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
version="1.0">
  <step id="batchlet-step">
    <batchlet ref=”com.trivadis.batch.HelloWorldBatchlet" />
  </step>
</job>
```

trivadis
makes IT easier.

JSR-352: Overview of job executions in Glassfish 4



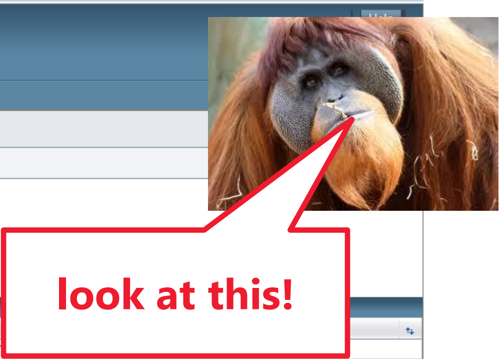
Batch Job Executions

View batch jobs that have been executed. Click an execution ID for details about a specific execution.

Instance Name: server

Batch Jobs (3)

Execution ID	Job Name	Batch Status	Exit Status	Instance ID	Start Time	
14	load-job-jsr352	COMPLETED	COMPLETED	13	2013-09-19 23:37	
13	load-job-jsr352	COMPLETED	COMPLETED	12	2013-09-19 23:37:47 CEST	2013-09-19 23:37:47 CEST
12	load-job-jsr352	COMPLETED	COMPLETED	11	2013-09-19 23:37:46 CEST	2013-09-19 23:37:46 CEST



trivadis
makes IT easier.

Batch Applications for the Java Platform (JSR-352)

Java Specification Request	JSR-352 (Version 1.0)
Reference Implementation	JBatch (https://java.net/projects/jbatch)
API: number of interfaces	~30
API: number of classes	~25 (~11 Exceptions)
Specification Lead	Chris Vignola (IBM)

part of Glassfish 4.0

Support	**Spring Batch**	**JSR-352**
File reading	Yes	No
Database reading	Yes	No
Admin interface available	Yes (Spring Batch Admin)	No (overview of executions so far)
Job Scheduling	No	No

Benchmarks

Number of periodic jobs	> 800 (per day)
Number of file loads	~ 8 (per day)
Number of data migrations	~ 15 (since june)
Average number of items per migration	~ 300 000
Items per second (single threaded)	~30*
Items per second (5 threads)	~150*

** in case of low error rate*

Questions?

THANK YOU.

Trivadis AG

Europa-Strasse 5
8152 Glattbrugg

michael.beer@trivadis.com
raffael.schmid@trivadis.com

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN

List of references

Tutorial: Create Batch Application based on JSR-352	http://www.planetjones.co.uk/blog/25-05-2013/introducing-jsr-352-java-batch-ee-7
Similarities and differences: Spring Batch vs. JSR-352	http://blog.codecentric.de