



Business  
Technology Days

BIG  
DATA  
CON

Dennis Schulte / Tobias Flohre | codecentric AG

# Enterprise Java Batch mit Spring

---

Dennis Schulte

codecentric 

Düsseldorf

@denschu

[www.github.com/denschu](http://www.github.com/denschu)

[blog.codecentric.de/author/dsc](http://blog.codecentric.de/author/dsc)

tel +49 (0) 1515 \_ 288 2395

[dennis.schulte@codecentric.de](mailto:dennis.schulte@codecentric.de)

[www.codecentric.de](http://www.codecentric.de)



---

Tobias Flohre

codecentric 

Düsseldorf

@TobiasFlohre

[www.github.com/tobiasflohre](http://www.github.com/tobiasflohre)

[blog.codecentric.de/author/tobias.flohre](http://blog.codecentric.de/author/tobias.flohre)

tel +49 (0) 162.2943073

[tobias.flohre@codecentric.de](mailto:tobias.flohre@codecentric.de)

[www.codecentric.de](http://www.codecentric.de)



# AGENDA

---

- Grundlagen Spring Batch
- Enterprise Java Batch
- Neuigkeiten im Bereich Java Batch

# AGENDA

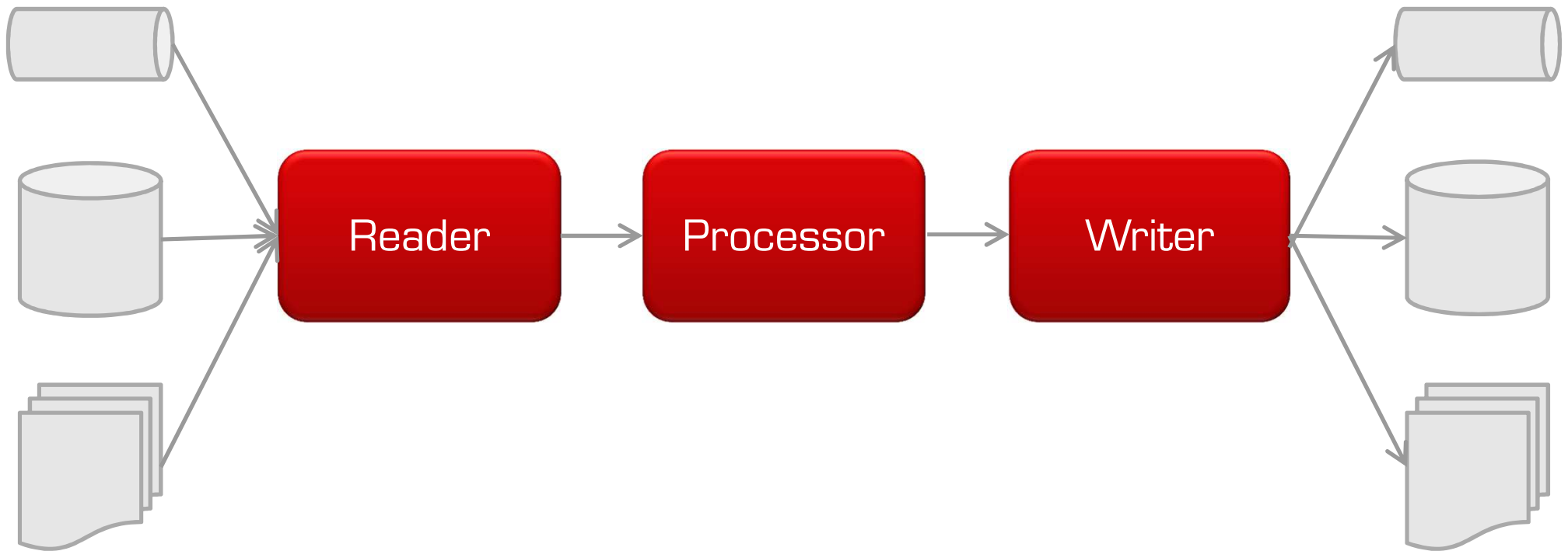
---

- **Grundlagen Spring Batch**
  - **Was ist ein Batch?**
  - Domain / Konfiguration / Ablauf
  - Restart / Skip / Listener
- Enterprise Java Batch
- Neuigkeiten im Bereich Java Batch

# WAS IST EIN BATCH?

---

## Traditionelles Batch-Pattern



# SPRING BATCH

---



**Restart**

**Automatisches  
Transaktionsmanagement**

**Persistente  
Job-Metadaten**

**Skip**

**Retry**

**Skalierungsfeatures**

# AGENDA

---

- **Grundlagen Spring Batch**
  - Was ist ein Batch?
  - **Domain / Konfiguration / Ablauf**
  - Restart / Skip / Listener
- Enterprise Java Batch
- Neuigkeiten im Bereich Java Batch



# DOMAIN / KONFIGURATION / ABLAUF

---

**Job**

**Step**

**Item**

**ItemReader**

**ItemWriter**

**ItemProcessor**

**Chunk**

# DOMAIN / KONFIGURATION / ABLAUF

---

- Job wird als Spring-Konfiguration erstellt
- Domain Specific Language manifestiert sich in XML-  
Namespace

- Zentrale Elemente

- job
- step
- tasklet
- chunk
  - reader
  - processor
  - writer
  - commit-interval

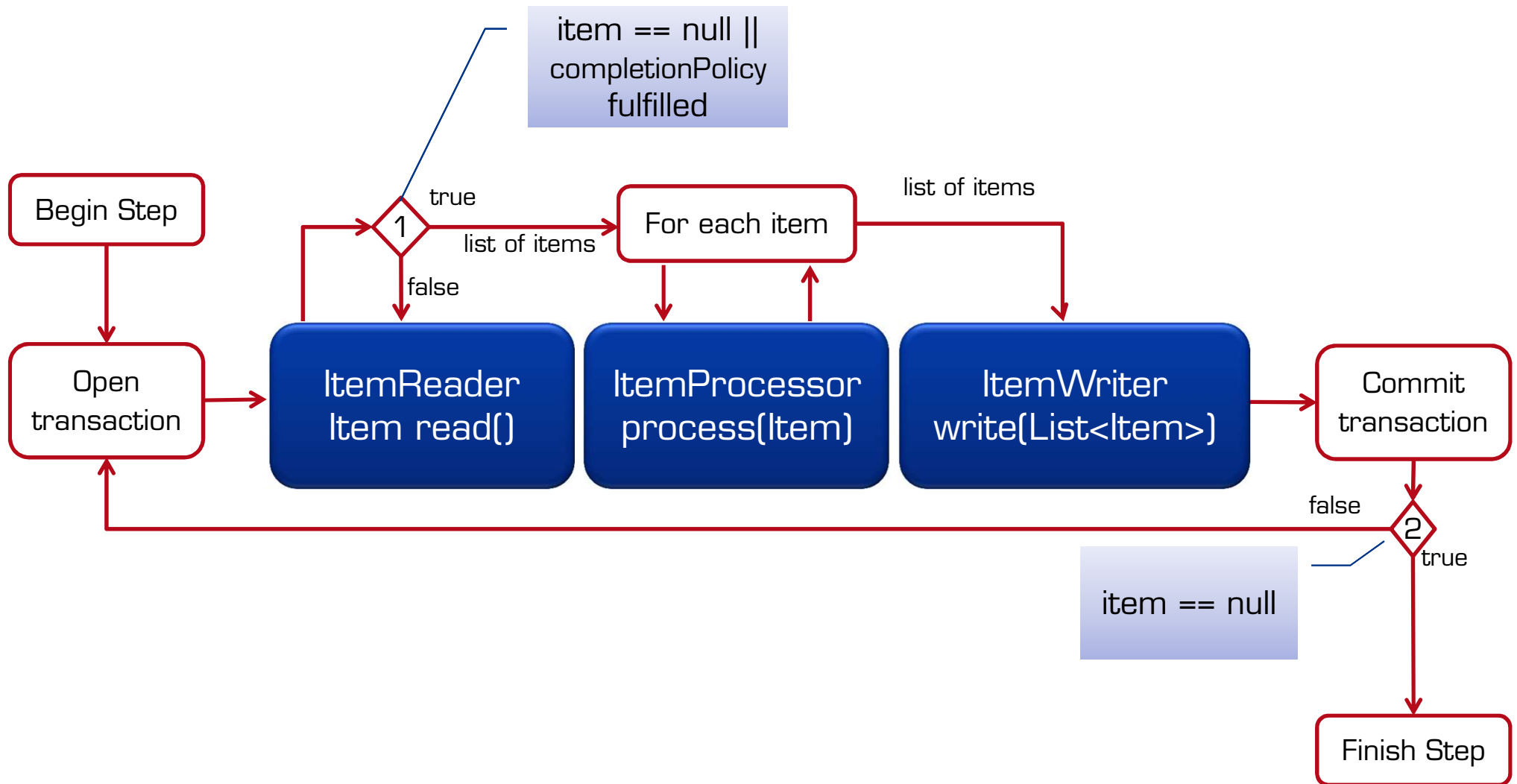
```
<job id="myJob" >  
  <step id="myStep" >  
    <tasklet>  
      <chunk reader="myReader"  
        processor="myProcessor" writer="myWriter"  
        commit-interval="1" />  
    </tasklet>  
  </step>  
</job>
```

# DOMAIN / KONFIGURATION / ABLAUF

---

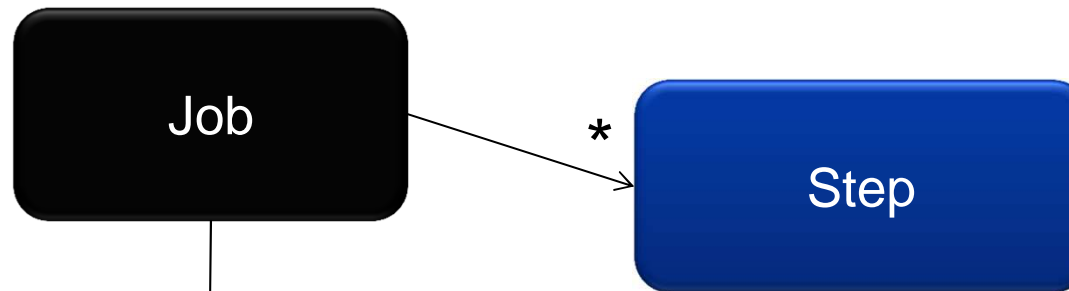
- Reader, Processor und Writer implementieren bestimmte Interfaces
  - ItemReader<T>
    - T read()
  - ItemProcessor<I,O>
    - O process(I item)
  - ItemWriter<T>
    - void write(List<? extends T> items)
- Spring Batch bietet für sehr viele Use Cases Implementierungen an
  - Lesen/Schreiben aus/in eine Datenbank
  - Lesen/Schreiben aus/in ein Flat-File
  - Lesen/Schreiben aus/in ein XML-File
  - Lesen/Schreiben aus/in eine JMS-Queue
  - Lesen/Schreiben mit JPA
  - und viele mehr

# DOMAIN / KONFIGURATION / ABLAUF

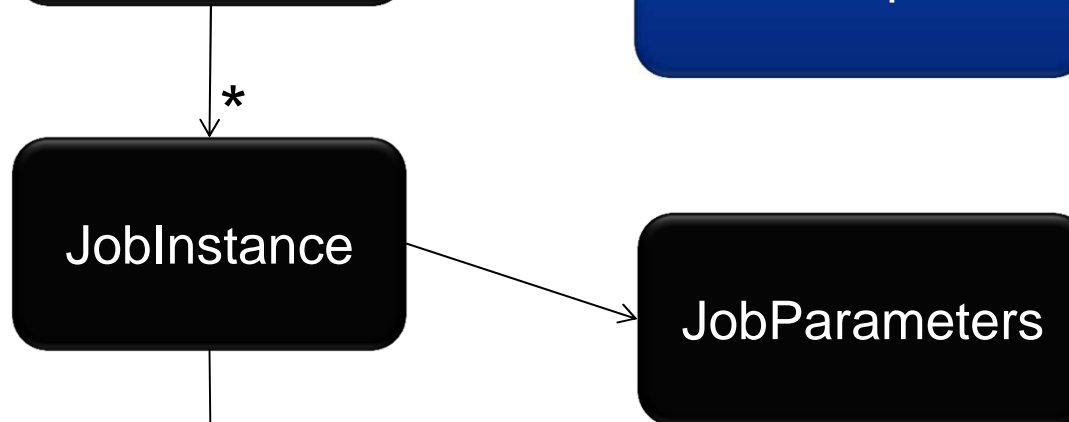


# DOMAIN / KONFIGURATION / ABLAUF

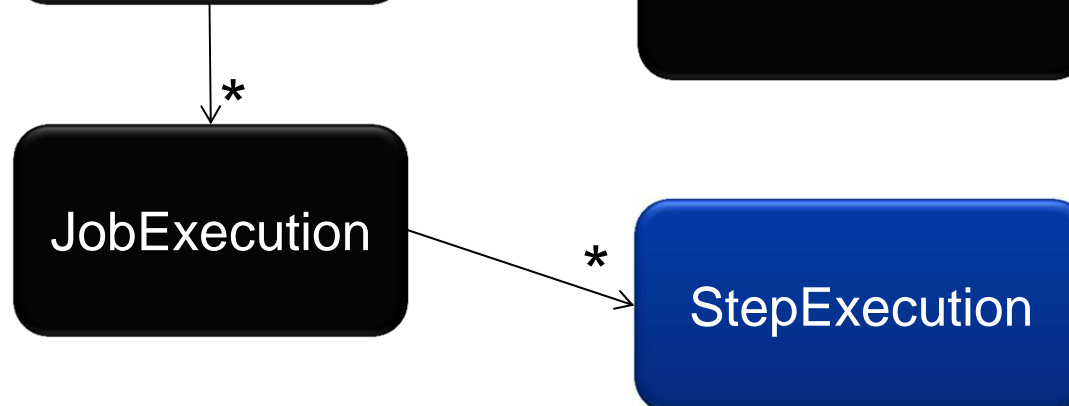
Inkasso  
Kraftfahrt



Inkasso  
Kraftfahrt am  
22.03.13



Inkasso  
Kraftfahrt am  
22.03.13  
erster Versuch



# DOMAIN / KONFIGURATION / ABLAUF

---

## – Infrastrukturkomponenten

JobLauncher

PlatformTransaction  
Manager

JobRepository

# AGENDA

---

- **Grundlagen Spring Batch**
  - Was ist ein Batch?
  - Domain / Konfiguration / Ablauf
  - **Restart / Skip / Listener**
- Enterprise Java Batch
- Neuigkeiten im Bereich Java Batch

# RESTART / SKIP / LISTENER

## Start des Jobs

JobInstance mit JobParams  
existiert nicht -> wird angelegt

JobExecution läuft los

Fehler!

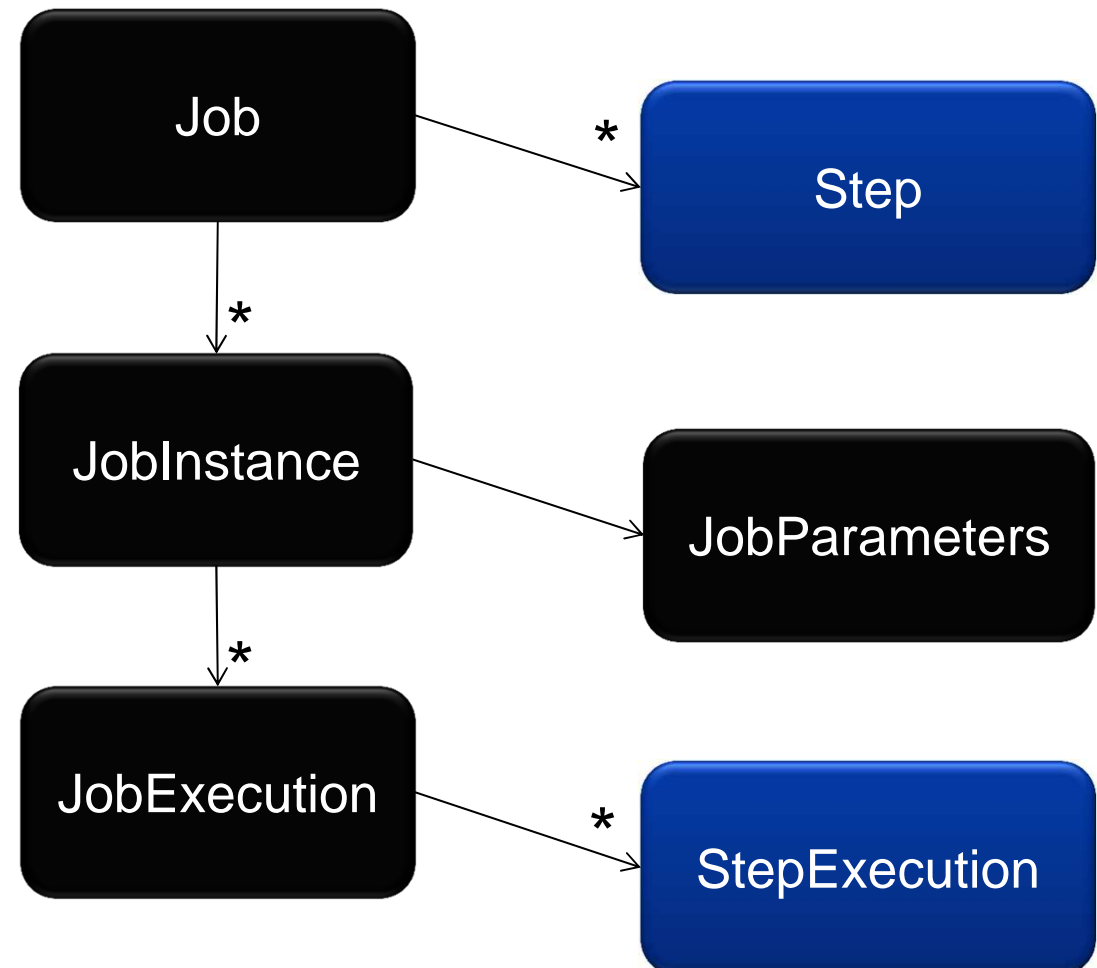
StepExecution, JobExecution und  
JobInstance erhalten Status

## Restart des Jobs

JobInstance mit JobParams  
existiert -> Status?

Neue JobExecution wird erzeugt,  
Zugriff auf ExecutionContext

Lauf erfolgreich





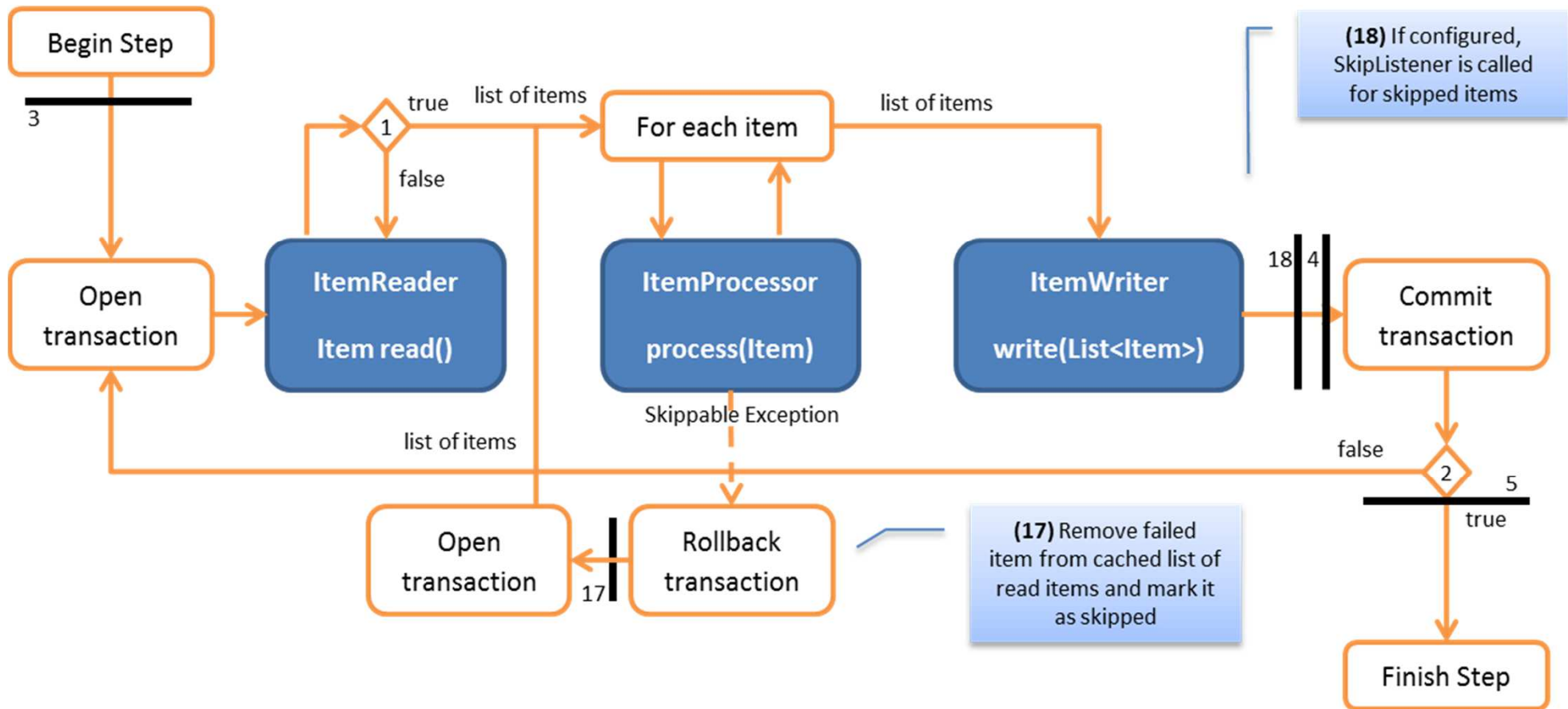
# RESTART / SKIP / LISTENER

---

- Überspringen fehlerhafter Elemente

```
<job id="chunkJob">
  <step id="chunkStep">
    <tasklet>
      <chunk reader="reader" processor="processor" writer="writer" commit-interval="5"
        skip-limit="20">
        <skippable-exception-classes>
          <include class="java.lang.IllegalArgumentException"/>
        </skippable-exception-classes>
      </chunk>
    </tasklet>
  </step>
</job>
```

# RESTART / SKIP / LISTENER



# RESTART / SKIP / LISTENER

---

- Listener horchen auf bestimmte Events im Batch-Ablauf
- Verschiedene Typen

- JobExecutionListener
- StepExecutionListener
- ChunkListener
- ItemReadListener
- ItemProcessListener
- ItemWriteListener
- SkipListener

```
<job id="chunkJob">
  <listeners>
    <listener ref="myJobExecutionListener"/>
  </listeners>
  <step id="chunkStep">
    <tasklet>
      <chunk reader="reader" processor="processor"
        writer="writer" commit-interval="5"/>
    </tasklet>
    <listeners>
      <listener ref="myStepListener"/>
    </listeners>
  </step>
</job>
```

# AGENDA

---

- Grundlagen Spring Batch
- Enterprise Java Batch
  - **Laufzeitumgebung / Integration von Fremdsystemen**
  - Default-Funktionalität
  - Test von Jobketten
- Neuigkeiten im Bereich Java Batch

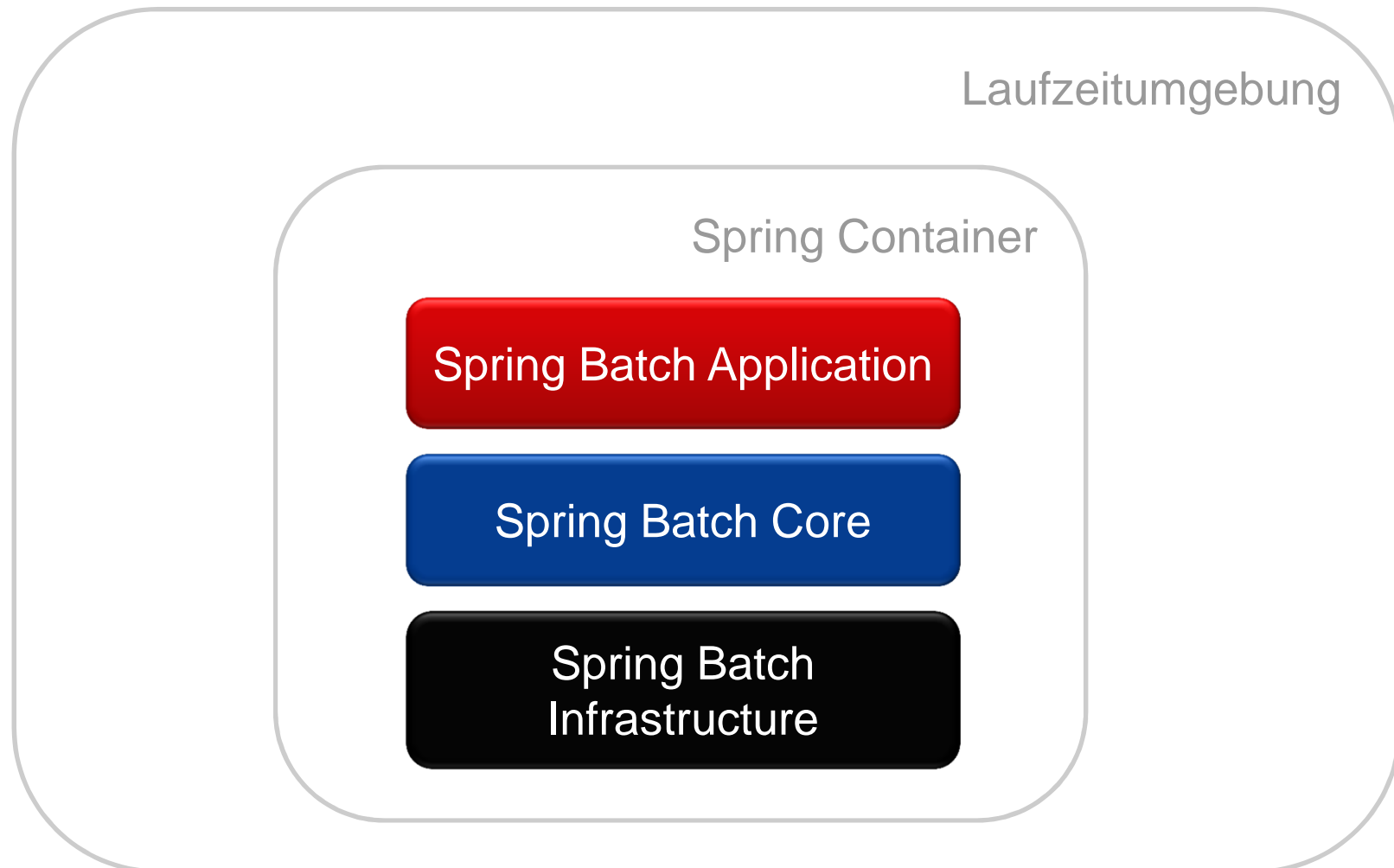
# LAUFZEITUMGEBUNG / INTEGRATION FREMDSYSTEME

---

- Spring Batch schreibt keine Laufzeitumgebung vor
  - Anwendungsframework!
  - Viele Deploymentvarianten denkbar
- Wichtige Fragen
  - Werden XA-Transaktionen benötigt?
  - Wie fügt sich die Ausführung des Jobs in einen Prozess ein?
  - In welcher Form und wo wird der Output benötigt?
  - In welcher Form soll ein Monitoring stattfinden?

# LAUFZEITUMGEBUNG / INTEGRATION FREMDSYSTEME

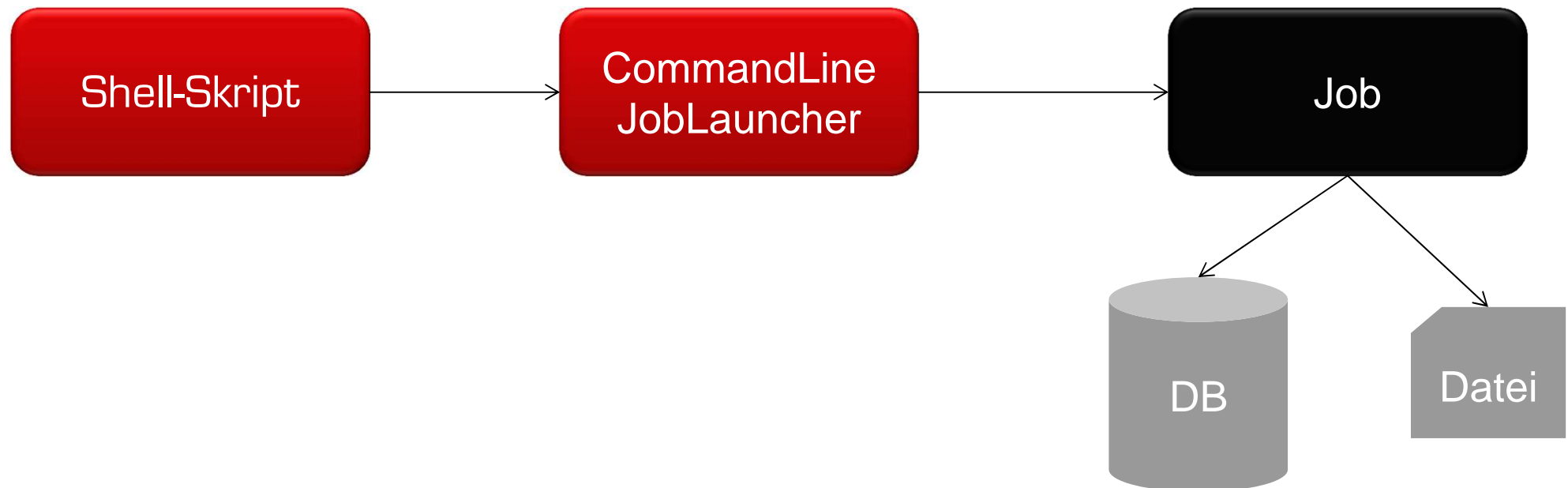
---



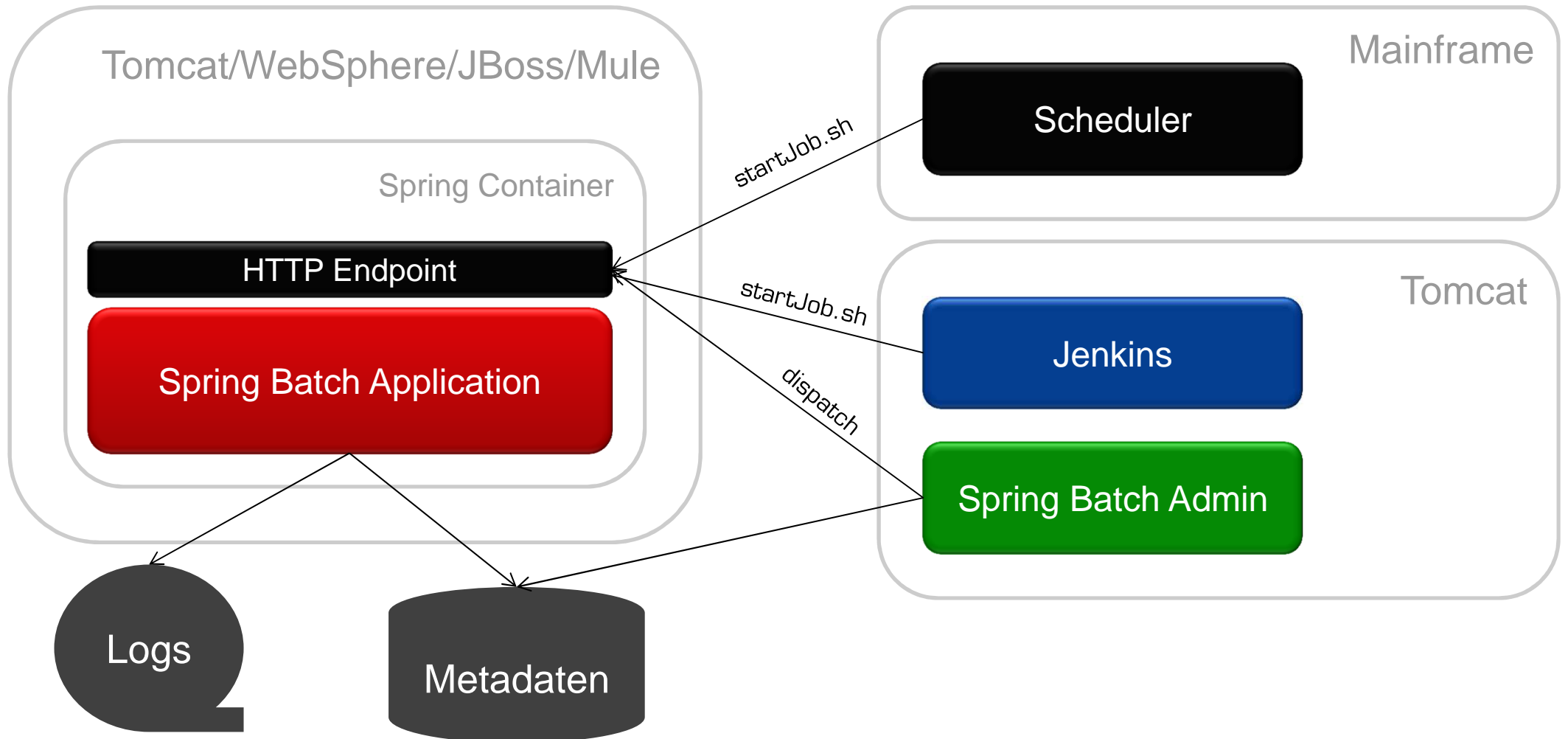
# LAUFZEITUMGEBUNG / INTEGRATION FREMDSYSTEME

- Standalone, 1 JVM pro Batch

```
java org.springframework.batch.core.launch.support.CommandLineJobRunner testJob.xml testJob date=2012/08/29
```



# LAUFZEITUMGEBUNG / INTEGRATION FREMDSYSTEME





# SPRING BATCH ADMIN



The screenshot shows the Spring Batch Admin web application. At the top, there is a green header with the text "Spring Batch Admin" and the SpringSource logo. Below the header is a navigation bar with links for "Home", "Jobs", "Executions", "SpringSource", and "Spring Batch". The main content area is titled "Job Names Registered" and contains a table with the following data:

Name	Description	Execution Count	Launchable
<a href="#">infinite</a>	No description	0	true
<a href="#">job1</a>	No description	0	true
<a href="#">job2</a>	No description	0	true

Below the table, it says "Rows: 1-3 of 3 Page Size: 20". At the bottom of the page, there is a footer with the text "© Copyright 2008 SpringSource. All Rights Reserved." and a link "Contact SpringSource".

# SPRING BATCH ADMIN

## Spring Batch Admin



Home Jobs Executions SpringSource Spring Batch

### Recent and Current Job Executions

ID	Instance	Name	Date	Start	Duration	Status	ExitCode	Steps
<u>3</u>	3	job1	2012-05-21	06:02:05	00:00:00	COMPLETED	COMPLETED	
<u>2</u>	2	job1	2012-05-21	06:01:52	00:00:00	FAILED	FAILED	
<u>1</u>	1	infinite	2012-05-21	06:01:33	00:00:04	FAILED	FAILED	
<u>0</u>	0	job1	2012-05-21	06:01:10	00:00:00	COMPLETED	COMPLETED	

Rows: 1-4 of 4 Page Size: 20

© Copyright 2008 SpringSource. All Rights Reserved. [Contact SpringSource](#)

# SPRING BATCH ADMIN

## Details for Job Execution

Stop

Property	Value
ID	3
Job Name	<u>job1</u>
Job Instance	<u>3</u>
Job Parameters	run.count(long)=2,fail=false
Start Date	2012-05-21
Duration	00:00:00
Status	COMPLETED
Exit Code	COMPLETED
Step Executions Count	1
<u>Step Executions</u>	<u>[job1.i1.s1]</u>

## History of Step Execution for Step=job1.j1.s1

Summary after total of 3 executions:

Property	Min	Max	Mean	Sigma
Duration	18	124	70,667	43,277
Commits	0	6	4	2,828
Rollbacks	0	1	0,333	0,471
Reads	1	5	3,667	1,886
Writes	0	5	3,333	2,357
Filters	0	0	0	0
Read Skips	0	0	0	0
Write Skips	0	0	0	0
Process Skips	0	0	0	0

## Details for Step Execution

Property	Value
ID	102
Job Execution	<u>3</u>
Job Name	job1
Step Name	job1.j1.s1
Start Date	2012-05-21
Duration	00:00:00
Status	COMPLETED
Reads	5
Writes	5
Filters	0
Read Skips	0
Write Skips	0
Process Skips	0
Commits	6
Rollbacks	0
Exit Code	COMPLETED
Exit Message	

# LAUFZEITUMGEBUNG / INTEGRATION FREMDSYSTEME

---

## – HTTP Endpoint mit Spring Integration

```
<int-http:inbound-gateway request-channel="job-requests" path="/jobs/{jobName}" supported-methods="POST">
    <int-http:header name="jobName" expression="#pathVariables.jobName"/>
</int-http:inbound-gateway>

<int:channel id="job-requests"/>

<int:service-activator input-channel="job-requests"
    expression="@messageHandler.launch(headers.jobName, payload.jobParameters)" />

<bean id="messageHandler" class="de.codecentric.batch.integration.JobLaunchingMessageHandler">
    <constructor-arg ref="jobOperator" />
</bean>
```

→ curl -s -data-urlencode "jobParameters=datum=01.01.2013"  
http://localhost:8080/batch/jobs/testJob

# AGENDA

---

- Grundlagen Spring Batch
- Enterprise Java Batch
  - Laufzeitumgebung / Integration von Fremdsystemen
  - **Default-Funktionalität**
  - Test von Jobketten
- Neuigkeiten im Bereich Java Batch

# DEFAULT-FUNKTIONALITÄT

---

- Vorgaben für Entwickler
  - Ressourcen
    - JobRepository
    - PlatformTransactionManager
    - DataSources
    - JMS-ConnectionFactory
    - Abstrakte JMSTemplate-Definitionen
    - Test nicht vergessen -> SpringJUnit4ClassRunner

```
@Test
```

```
public void testLaunchJob() throws Exception {
```

```
    JobParameters jobParameters = new JobParametersBuilder().toJobParameters();
```

```
    JobExecution jobExecution = jobLauncher.run(job, jobParameters);
```

```
    assertEquals(BatchStatus.COMPLETED, jobExecution.getStatus());
```

```
}
```

# DEFAULT-FUNKTIONALITÄT

---

## — Abstrakte Job- und Stepdefinitionen

```
<job id="abstractJob" abstract="true" >
  <listeners>
    <listener ref="myDefaultJobExecutionListener"/>
  </listeners>
</job>
```

```
<step id="abstractStep" abstract="true">
  <tasklet>
    <listeners>
      <listener ref="myDefaultStepExecutionListener"/>
    </listeners>
  </tasklet>
</step>
```

```
<job id="chunkJob" parent="abstractJob">
  <listeners merge="true">
    <listener ref="myJobExecutionListener"/>
  </listeners>
  <step id="chunkStep" parent="abstractStep">
    <tasklet>
      <chunk reader="reader" processor="processor"
        writer="writer" commit-interval="5"/>
      <listeners merge="true">
        <listener ref="myStepListener"/>
      </listeners>
    </tasklet>
  </step>
</job>
```

# DEFAULT-FUNKTIONALITÄT

---

- Log-Files pro Job-Lauf
  - Mapped Diagnostic Context -> Log-File-Name (thread-local)
- Format und Ausgabe des Job-Protokolls
  - Job-Steuerung
  - Revisionssichere Speicherung
- Mapping von Return-Codes
  - Scheduler erwartet bestimmte Codes



# AGENDA

---

- Grundlagen Spring Batch
- Enterprise Java Batch
  - Laufzeitumgebung / Integration von Fremdsystemen
  - Default-Funktionalität
  - **Test von Jobketten**
- Neuigkeiten im Bereich Java Batch

# CONTINUOUS DELIVERY

---

**Build → Release → Deploy → Test → Run**

# JOB PIPELINE

The screenshot displays the Jenkins Job Pipeline interface. At the top left, the Jenkins logo and 'Jenkins » Job Pipeline' are visible. A search bar and user information 'hv14x98 | Abmelden' are on the top right. Below the header, a 'Build Pipeline' section features a cartoon character and a toolbar with icons for Run, History, Configure, Add Step, Delete, and Manage. The pipeline itself is a horizontal sequence of five job blocks: 'Pipeline version 1' (No parameters), 'api-server-job' (Apr 2, 2013 8:38:26 AM, 1, 82 ms, hv14x98), 'file-transfer-job' (Apr 2, 2013 8:38:31 AM, 1, 27 ms), 'transformation-job' (Apr 2, 2013 8:38:36 AM, 1, 23 ms), and 'archiv-job' (Apr 2, 2013 8:38:41 AM, 1, 31 ms). Green arrows indicate the flow between jobs. At the bottom, there is a link to localize the page and the page creation date '02.04.2013 08:40:07' along with 'Jenkins ver. 1.451'.

# JOB PIPELINE

---



# JOB PIPELINE



# AGENDA

---

- Grundlagen Spring Batch
- Enterprise Java Batch
- **Neuigkeiten im Bereich Java Batch**
  - **Spring Batch 2.2**
  - JSR 352: Java Batch
  - Spring Batch und Hadoop

# SPRING BATCH 2.2

---

- In Entwicklung (bisher Release Candidate 1)
- Bugfixes und neue Features
- AmqpItemReader / AmqpItemWriter
- JobParameter bestimmen nicht zwangsläufig Identität
- Unterstützung für Java-basierte Konfiguration

# SPRING BATCH 2.2

---

- Java-basierte Konfiguration (JavaConfig)
- XML-basierte Konfiguration

```
@EnableTransactionManagement
```

```
@Configuration
```

```
public class MyConfiguration {
```

```
    @Bean
```

```
    public MyComponent component(){
```

```
        return new MyComponent();
```

```
    }
```

```
}
```

```
<beans ...>
```

```
    <tx:annotation-driven/>
```

```
    <bean id="component"
```

```
        class="de.codecentric.MyComponent"/>
```

```
</beans>
```



# SPRING BATCH 2.2

---

- Neue Annotation: `@EnableBatchProcessing`
- Benötigt eine `DataSource` und erzeugt
  - `JobRepository`, `JobLauncher`, `JobRegistry`
  - `DataSourceTransactionManager`
  - `JobBuilderFactory`, `StepBuilderFactory`

```
@Configuration
```

```
@EnableBatchProcessing
```

```
public class InfrastructureConfiguration {
```

```
    @Bean
```

```
    public DataSource dataSource(){
```

```
        EmbeddedDatabaseBuilder embeddedDatabaseBuilder = new EmbeddedDatabaseBuilder();
```

```
        return embeddedDatabaseBuilder.addScript("../schema-hsqldb.sql").setType(EmbeddedDatabaseType.HSQL)  
            .build();
```

```
    }
```

```
}
```

# SPRING BATCH 2.2

---

@Configuration

```
public class JobConfiguration {
```

```
    @Autowired
```

```
    private JobBuilderFactory jobBuilderFactory;
```

```
    @Autowired
```

```
    private StepBuilderFactory stepBuilderFactory;
```

```
    @Bean
```

```
    public Job job(){
```

```
        return jobBuilderFactory.get("job").start(step()).build();
```

```
    }
```

```
    @Bean
```

```
    public Step step(){
```

```
        return stepBuilderFactory.get("step").chunk(1).reader(reader()).
```

```
            processor(processor()).writer(writer()).build();
```

```
    }
```

```
    ...
```

```
}
```

# AGENDA

---

- Grundlagen Spring Batch
- Enterprise Java Batch
- **Neuigkeiten im Bereich Java Batch**
  - Spring Batch 2.2
  - **JSR 352: Java Batch**
  - Spring Batch und Hadoop

# JSR 352: JAVA BATCH

---

- JSR 352 = Java Specification Request für Java Batch
- Haupttreiber sind IBM/VMWare
  - Chris Vignola: Specification Lead (IBM)
  - Michael Minella, Wayne Lund (VMWare)
- Übernahme der Domain Language von Spring Batch in großen Teilen
- JSR definiert keine Infrastruktur

# JSR 352: JAVA BATCH

---

## – Interfaces vs. Annotations

```
public class MyItemReader implements  
ItemReader {  
  
    @Override  
    public T readItem() {...}  
  
}
```

```
public class MyItemReader {  
  
    @ReadItem  
    public T readItem() {...}  
  
}
```

# AGENDA

---

- Grundlagen Spring Batch
- Enterprise Java Batch
- **Neuigkeiten im Bereich Java Batch**
  - Spring Batch 2.2
  - JSR 352: Java Batch
  - **Spring Batch und Hadoop**

# SPRING BATCH UND HADOOP

---

- Spring Data: Apache Hadoop
  - Version 1.0 seit 26.02.2013
- Beinhaltet Tools, Templates und Co. für eine Integration zwischen Spring und Hadoop
- Hadoop Tasklet für Spring Batch
  - Einbindung von Hadoop-Jobs in Spring Batch - Jobketten

```
<batch:job id="springBatchJob" >
    <batch:step id="step">
        <batch:tasklet ref="myHadoopTasklet"/>
    </batch:step>
</batch:job>

<hdp:tasklet id="myHadoopTasklet" job-ref="myHadoopJob" wait-for-job="true"/>

<hdp:job id="myHadoopJob" mapper="de.codecentric.MyMapper" reducer="de.codecentric.MyReducer"
    input-path="/input" output-path="/output"/>
```

# FRAGEN?

---

Dennis Schulte / Tobias Flohre

codecentric AG  
Merscheider Straße 1  
42699 Solingen

[tobias.flohre@codecentric.de](mailto:tobias.flohre@codecentric.de)  
[dennis.schulte@codecentric.de](mailto:dennis.schulte@codecentric.de)

[www.codecentric.de](http://www.codecentric.de)  
[www.mbg-online.de](http://www.mbg-online.de)  
[blog.codecentric.de](http://blog.codecentric.de)  
[www.meettheexperts.de](http://www.meettheexperts.de)

