



Business
Technology Days

BIG
DATA
CON

Mark Struberg / INSO, TU Vienna

JBatch with Apache BatchEE

About the Speakers

- <http://github.com/struberg>
- <http://people.apache.org/~struberg>
- Apache Software Foundation member
- Apache OpenWebBeans, MyFaces, BVal, OpenJPA, Maven, DeltaSpike, BatchEE, ...
- Java EG member

Agenda

- JBatch and Apache BatchEE
- JBatch Terms and Ideas
- Writing a simple Batch
- Threading and Transactions
- JBatch Execution Environments
- BatchEE add-ons

About Apache BatchEE

- Apache Incubator Project since 2013-10
- Homepage:
<http://batchee.incubator.apache.org>
- Source:
<git://git.apache.org/incubator-batchee.git>
- dev@batchee.incubator.apache.org
- [irc.freenode.net](irc://irc.freenode.net) channel #apache-batchee
- <https://issues.apache.org/jira/browse/BATCHEE>

Diff JBatch and BatchEE

- JBatch
 - The specification
 - The RI
- BatchEE
 - includes forked and bug-fixed RI code for Java EE
 - adds tons of features on top of it:
 - different SPI implementations
 - various ItemReader and Writer,
 - JAX-RS and simple Job UI,
 - Standalone Java EE Batch Environment

JBatch Basics

What is JBatch

- A Java EE specification, JSR-352
 - <https://jcp.org/en/jsr/detail?id=352>
- Inspired by WebSphere-Batch, Spring-Batch, JES, JCL, ...
- To **write** and **execute** Java Batches
- Why do I need Batches ?
 - Migration
 - Bulk Import/Export
 - Mass processing

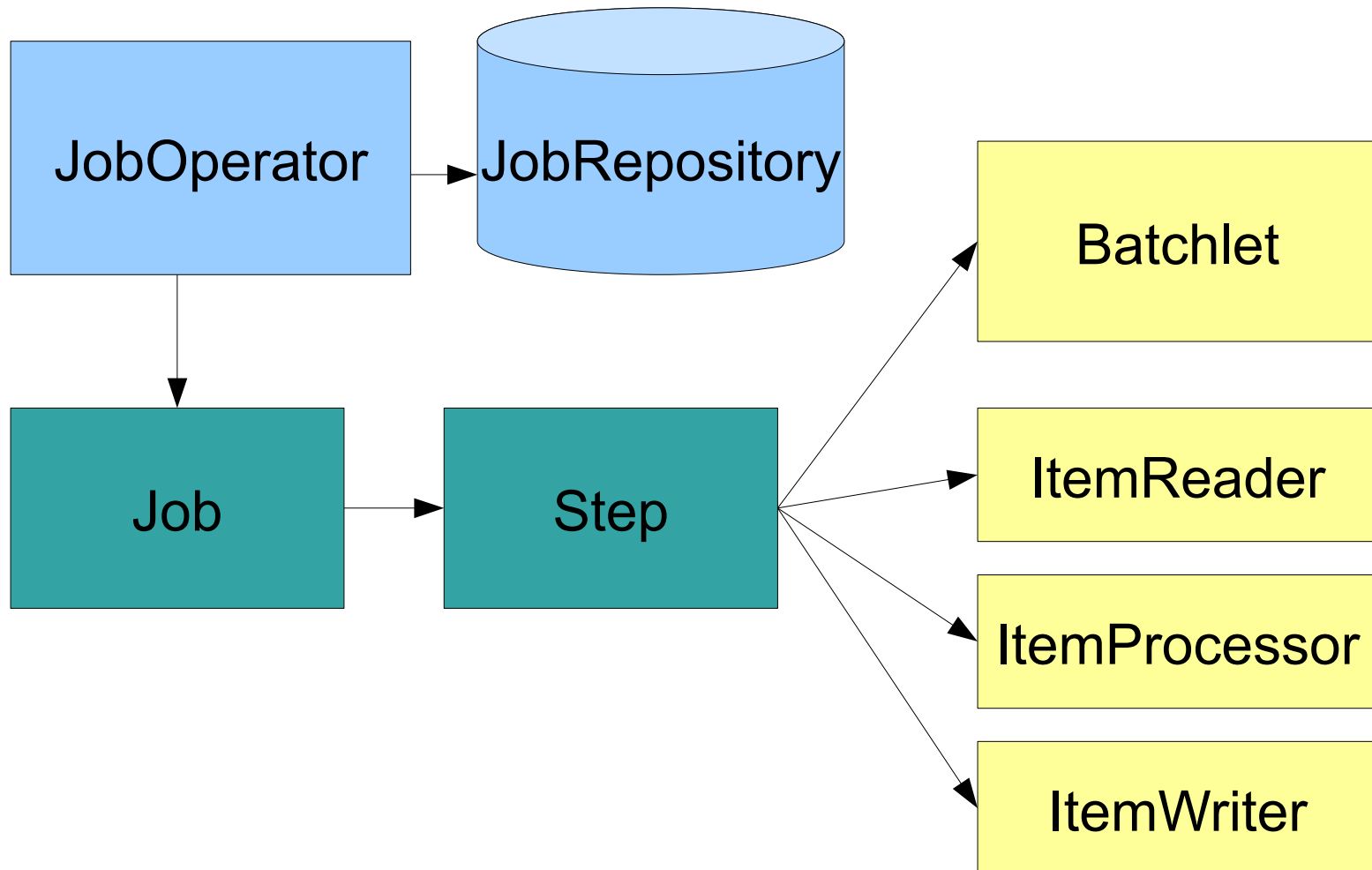
Batch Classification

- Online vs Offline Tasks
- Bulk vs On-demand Batches
- Synchronous vs Asynchronous processing

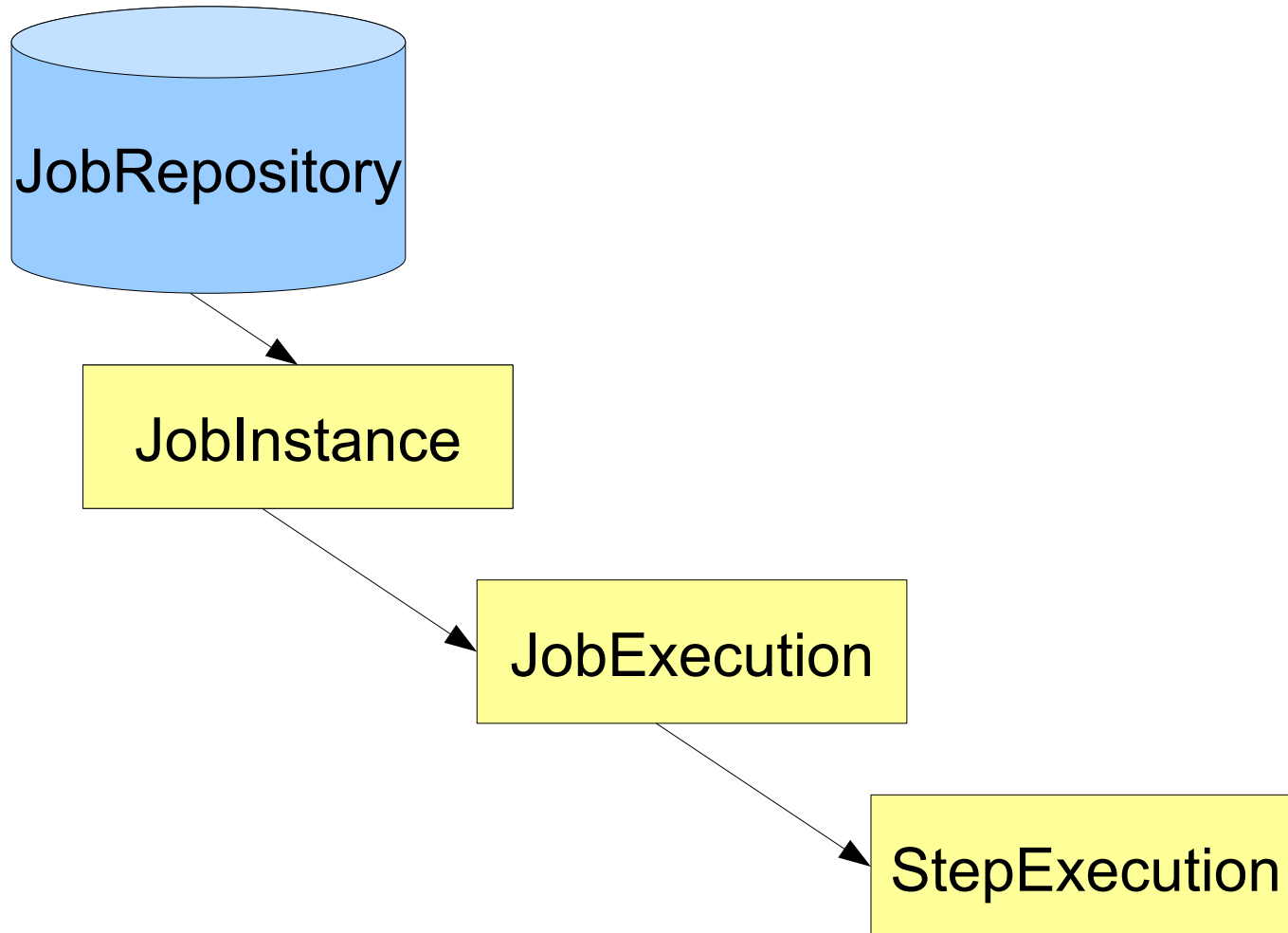
Benefits of JBatch Spec

- improved maintainability
- improved portability
- reuse of existing app components
- out-of-the-box monitoring and control

JBatch Architecture



JobRepository



Batch JSL

- JSL: Job Specification Language
- META-INF/batch-jobs/*.xml

```
<job id="myjob" restartable="false">  
  <step id="firststep">  
    <chunk item-count="1">  
      <reader ref="myReader"/>  
      <processor ref="myProcessor"/>  
      <writer ref="myWriter"/>  
    </chunk>  
  </step>  
</job>
```

Batch Artifact Loading

- Available as @Named
- Registered via META-INF/batch.xml

Batch Properties (1/2)

- Defined In JSL:

```
<properties>
```

```
  <property name="myProp" value="myVal" />
```

```
</properties>
```

- for:
 - <job>
 - <reader>, <processor>, <writer>
 - <batchlet>
- Dynamically via JobOperator

Batch Properties (2/2)

- Usage in Code via
 - `JobContext#getProperties()`;
 - `StepContext#getProperties()`;
 - `@Inject @BatchProperty("myProp") String val;`
- Can leverage Expression Language:
 - `<property name="csvFile"`
 - `value="#{jobProperties['infile.name']}" />`
 - `<property name="jobDay"`
 - `value="#{partitionPlan['schedule.day']}" />`

Basic Step Structure

- Batchlet or
- ItemReader, plus
- ItemProcessor (optional), plus
- ItemWriter
- Steps can form Sequence Flows
- Steps can use Partitioning and Splits
 - each Partition gets an own Thread

Batchlet

- performs all the processing in one go
 - `Batchlet#process()`
 - `Batchlet#stop()`
- Return the exit status of the batch

ItemReader

- Read single Item for a chunk step
- Can set and read 'Checkpoints'
- `readItem()` invoked until `null` returned

BatchEE TypedItemReader

- $\langle R, C \rangle$ version of ItemReader
 - R: Type of item to Read
 - C: Type of Checkpoint (? extends Serializable)
- $\langle R \rangle$ NoStateTypedItemReader
- $\langle R \rangle$ BufferedItemReader

ItemProcessor

- Optional!
- Processes the read item
- Returning `null` will skip the item

BatchEE TypedItemProcessor

- $\langle I, O \rangle$ version of ItemProcessor
 - I: Type of item input
 - O: Type of Output

ItemWriter

- Writes n items (according to chunk-size)
- Each writeItems is an own Transaction
- Can set and read 'Checkpoints'

BatchEE TypedItemWriter

- $\langle W, C \rangle$ version of ItemWriter
 - W: Type of item to Write
 - C: Type of Checkpoint (? extends Serializable)
- $\langle R \rangle$ NoStateTypedItemWriter

Batch Transaction Handling

- JBatch uses JTA by default
- JTA TX is always per Thread in Java
- Each 'Split' Partition has an own Transaction
- Commit packages can be set via chunk-size

Batch Chunks

- chunk-size=100 means
 - beginTransaction();
 - 100x ItemReader#readItem();
 - (optional) 100x ItemProcessor#processItem();
 - 1x ItemWriter#writeItems(all100);
 - commit();

Chunk Exception Handling

- default: abort w FAILED on Exception
- possible to define Exceptions for:
 - retry <retryable-exception-classes>
plus optionally: Retry*Listener
 - skip <skippable-exception-classes>
plus optionally: Skip*Listener
 - Retry is always done with item-count=1
 - Retry precedes Skip

The JobOperator

- THE central interface to communicate with the Batch Runtime
- can start, stop and restart Jobs
- can list Jobs

Batch Listener

- JobListener
 - `beforeJob()`
 - `afterJob()`
- StepListener
 - `beforeStep()`
 - `afterStep()`
- e.g. for setting up Authentication
- e.g. for starting CDI Contexts

BatchEE SPI

BatchEE SPI activation

- batchee.properties
 - `BatchThreadPoolService=....`
- SecurityService
- PersistenceManagerService
- BatchThreadPoolService
- TransactionManagementService
- <http://batchee.incubator.apache.org/configuration.html>

Batch Environments

Batch WAR

- BatchEE Servlet
- UI for
 - showing status
 - creating Jobs
 - starting Jobs

JAX-RS Integration

- Exposes JobOperator via JAX-RS
- Allows to start Jobs from UC4, JES, etc

CLI Batch

- We introduced our own BAR format

```
BATCH-INF
```

```
+-- batch-jobs/*.xml
```

```
+-- classes/..
```

```
+-- lib/..
```

- `$> java -jar batchee-cli.jar start \`
`-archive demo-1.0.bar -name mybatch`

CDI Batch Scopes

- @StepScoped
- @JobScoped
- technically done via StepListener and JobListener

Clustering

- TODO... it's just not yet done...
- ... helping hands are welcome...

Legal stuff

- Apache, BatchEE, OpenWebBeans, OpenEJB and OpenJPA are trademarks of the Apache Software Foundation