Wayne Lund

- Nov 8th 2007

Speakers' life history

- Wayne Lund is a Senior Architect for Accenture Delivery Architectures
  - Joined Accenture in 1992
  - Started working with Spring in 2004
  - Approached Rod Johnson @ Spring Experience in Dec 2005 about Spring Batch opportunity

- Other Key Spring Batch Contacts
  - Scott Wintermute is the Program Manager for Accenture Delivery Architectures
  - Lucas Ward is a Consultant with Accenture and co-technical lead of Spring Batch
  - Ben Hale is a Senior Consultant at Interface 21 and the new co-technical lead of Spring Batch
  - Dave Syer is a Principal Consultant with Interface21 and previous co-technical lead of Spring Batch still deeply involved

Overall presentation goal

How to implement reliable enterprise batch processing with Spring Batch

Agenda

- Java Batch – A Missing Enterprise Capability
- Objectives, Scenarios, and Features
- Spring Batch Overview
- Pseudo Code Scenarios
- Batch Domain Reference Model
- Demo
- Roadmap and Summary
- Q & A

Java Batch – A Missing Enterprise Capability in the Market

- Batch jobs are part of most IT projects and currently no commercial or open source framework provides a robust, enterprise-scale solution/framework
- Batch processing is an Application Style for data processing pipelines (e.g. payment and settlement systems)
- The lack of a standard architecture has led many projects to create their own custom architecture at significant development and maintenance costs

Spring Batch:
An Accenture and Interface21 Partnership

- Why is Accenture contributing to open source?
  - Consolidating decades worth of experience in building high-performance batch solutions
  - Driving standardization in batch processing
- Why Spring?
  - Established, active, and leading community with significant momentum
  - Logical home for a batch architecture framework as part of the Spring Portfolio
- End Goal
  - Provide a highly scalable, easy-to-use, customizable, industry-accepted Batch architecture framework

Agenda

- Java Batch – A Missing Enterprise Capability
- Objectives, Scenarios, and Features
- Spring Batch Overview
- Pseudo Code Scenarios
- Batch Domain Reference Model
- Demo
- Roadmap and Summary
- Q & A

Spring Batch: Technical and Architecture Objectives

- Clear separation of concerns – application developer concentrates on business logic
- Provide common, architecture layer services as interfaces
- Provide simple and default implementations of the layers that are easy to configure, customize, and extend
- Batch services should be easy to replace or extend, without impact to the infrastructure or application layers

Spring Batch: Business Scenarios

- Commit batch process periodically
- Concurrent batch processing: parallel processing of a jobs
- Manual or scheduled restart after failure
- Partial processing: skip records (e.g., on rollback)
- Sequential processing of dependent steps
- Staged, enterprise message-driven processing
- Whole-batch transaction for simple data models or small batch size
- Massively parallel batch processing

Spring Batch: Features

- Support for multiple file formats
  - fixed length, delimited, XML…
- Automatic retry after failure
- Job control language for monitoring and operations
  - start, stop, suspend, cancel
- Execution status and statistics during a run and after completion
- Multiple ways to launch a batch job
  - http, Unix script, incoming message, etc.
- Ability to run concurrently with OLTP systems
- Ability to use multiple transaction resources
- Support core batch services
  - logging, resource management, restart, skip, etc.

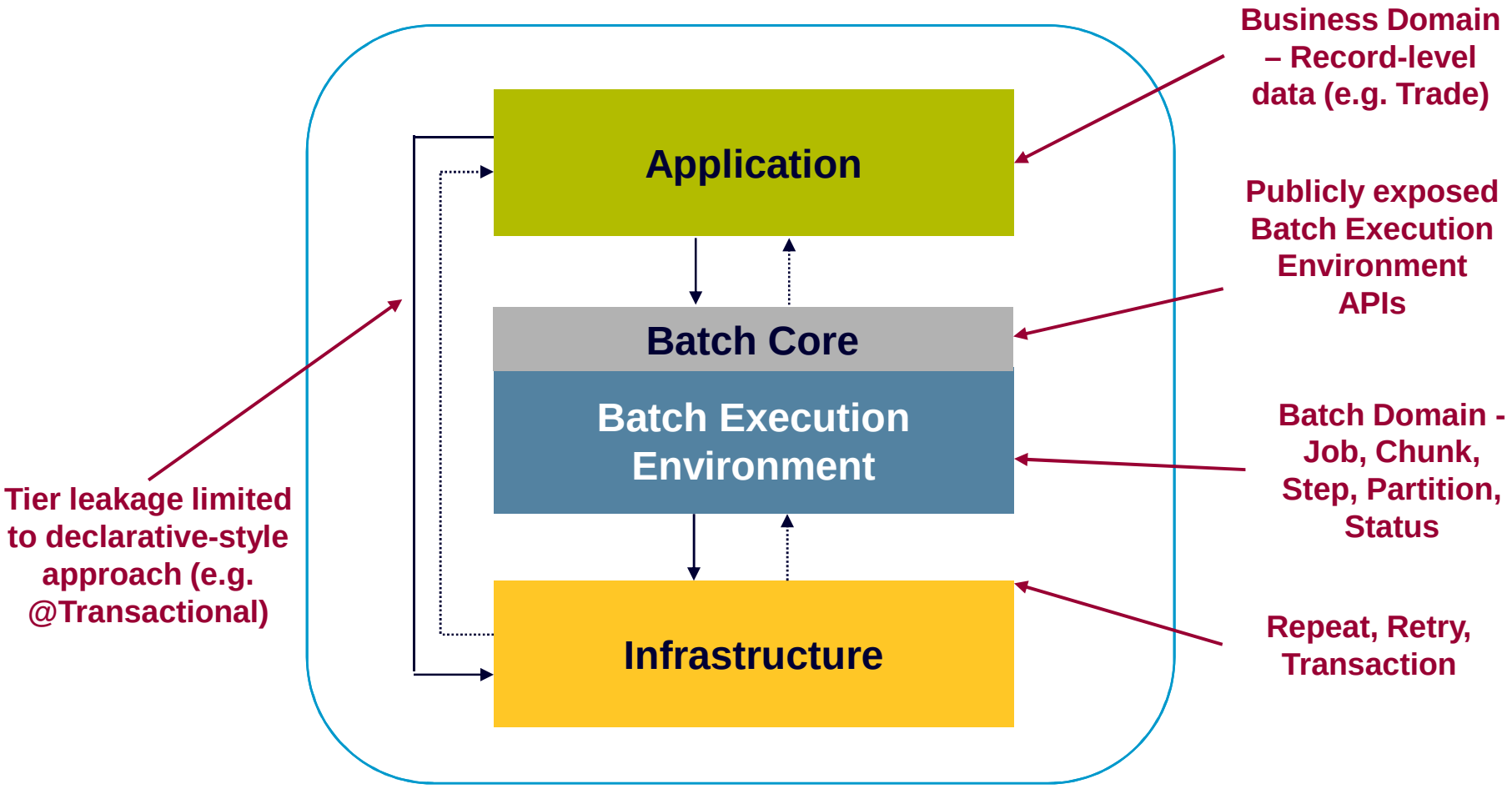
Agenda

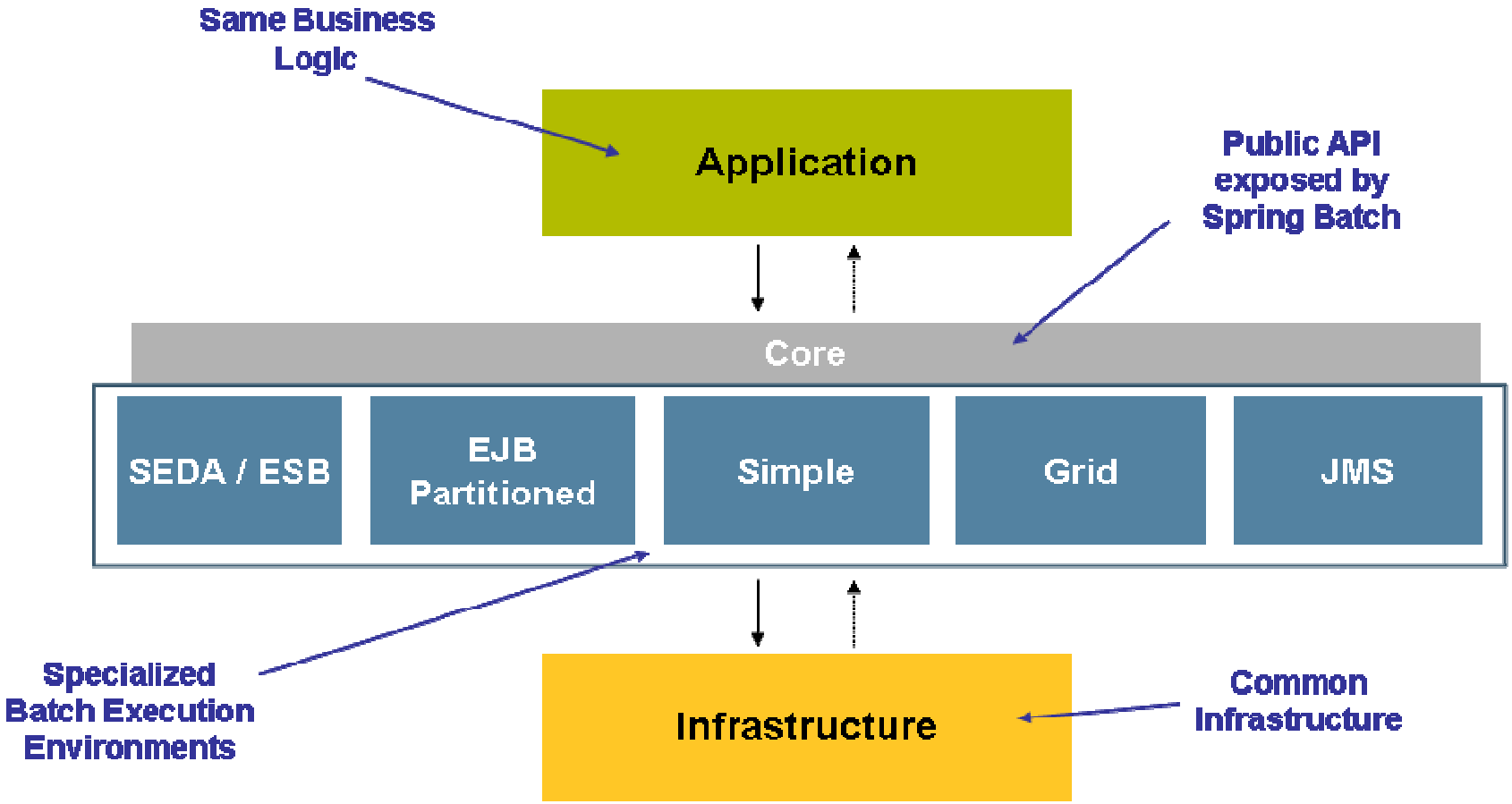
- Java Batch – A Missing Enterprise Capability
- Objectives, Scenarios, and Features
- Spring Batch Overview
- Pseudo Code Scenarios
- Batch Domain Reference Model
- Demo
- Roadmap and Summary
- Q & A

Spring Batch: Infrastructure

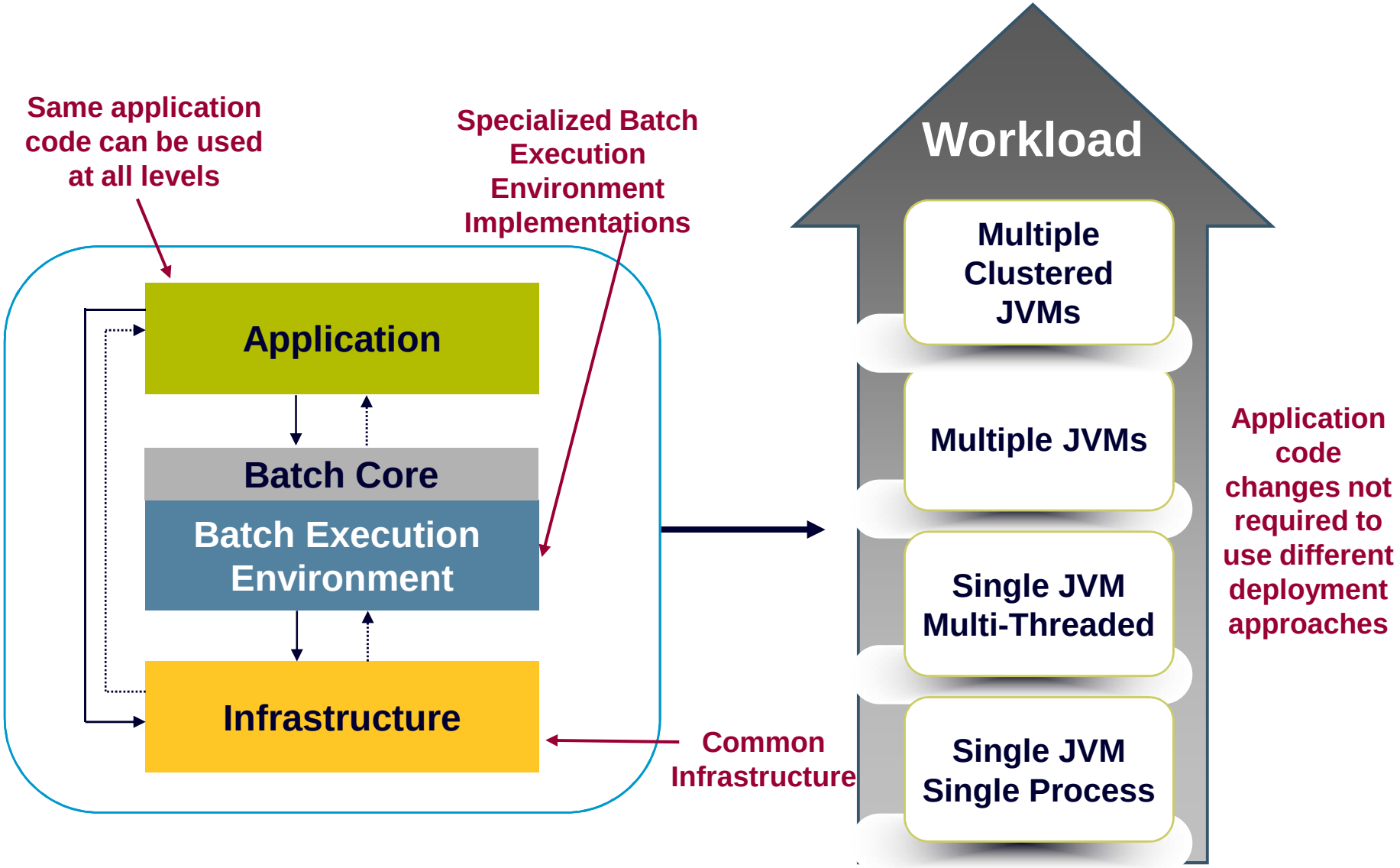
- All the business scenarios can be implemented in terms of lower level concepts
- All can be nested and composed together to build batch execution environment features
- Identify three such infrastructure concepts:
  - **Transaction** – atomic, consistent, isolated, durable
  - **Repeat** – automatically repeat an operation **unless** it fails, e.g. a fixed number of times
  - **Retry** – automatically retry an operation **until** it succeeds, e.g. no more input

Spring Batch: Layered Architecture





Spring Batch: Approaches to Scale



Agenda

- Java Batch – A Missing Enterprise Capability
- Objectives, Scenarios, and Features
- Spring Batch Overview
- Pseudo Code Scenarios
- Batch Domain Reference Model
- Demo
- Roadmap and Summary
- Q & A

Simple Batch Pseudo Code

```
REPEAT(while more input) {

      TX {
            REPEAT(size=500) {

                  input;
                  output;

            }
      }

}
```

RepeatTemplate

TransactionTemplate

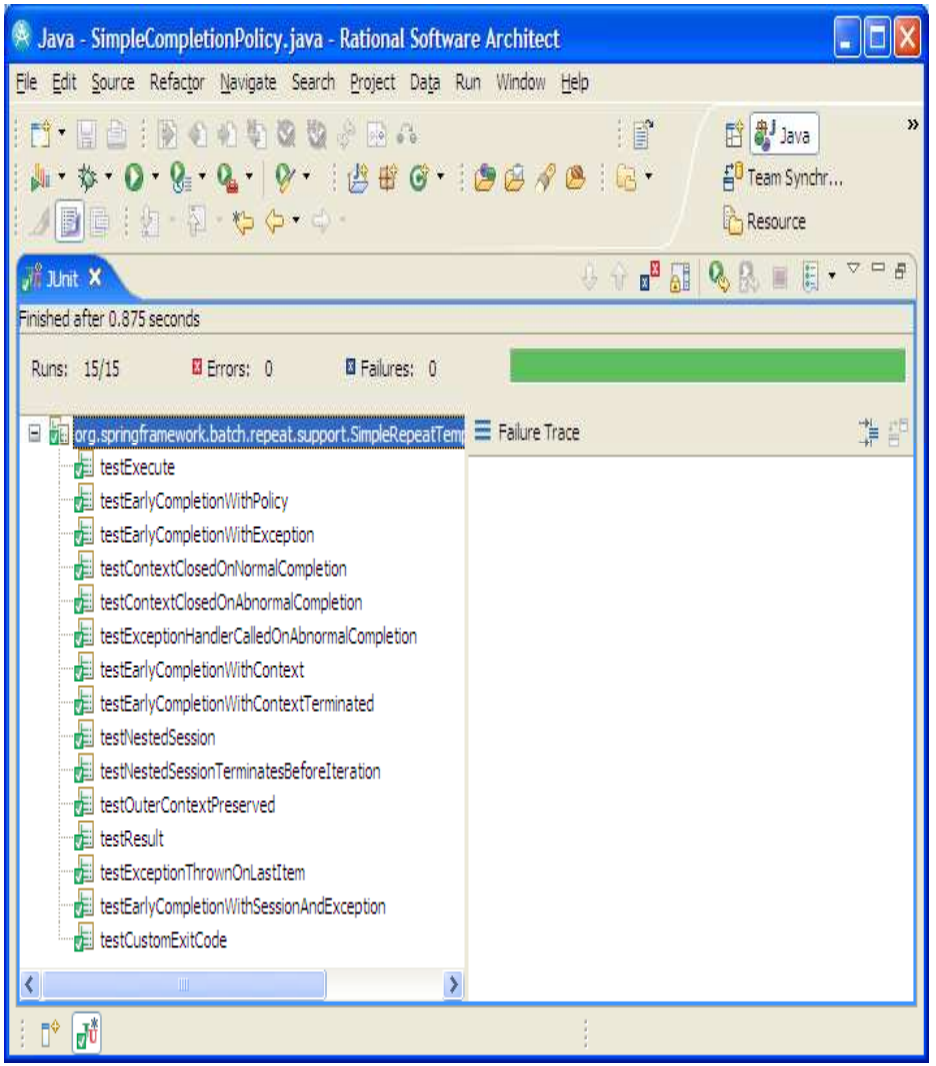
RepeatTemplate

Business Logic

Note: For reference, see Unit Test → Infrastructure: SimpleRepeatTemplateTests using trades.csv

Ensure Failed Transaction Does not Terminate Job

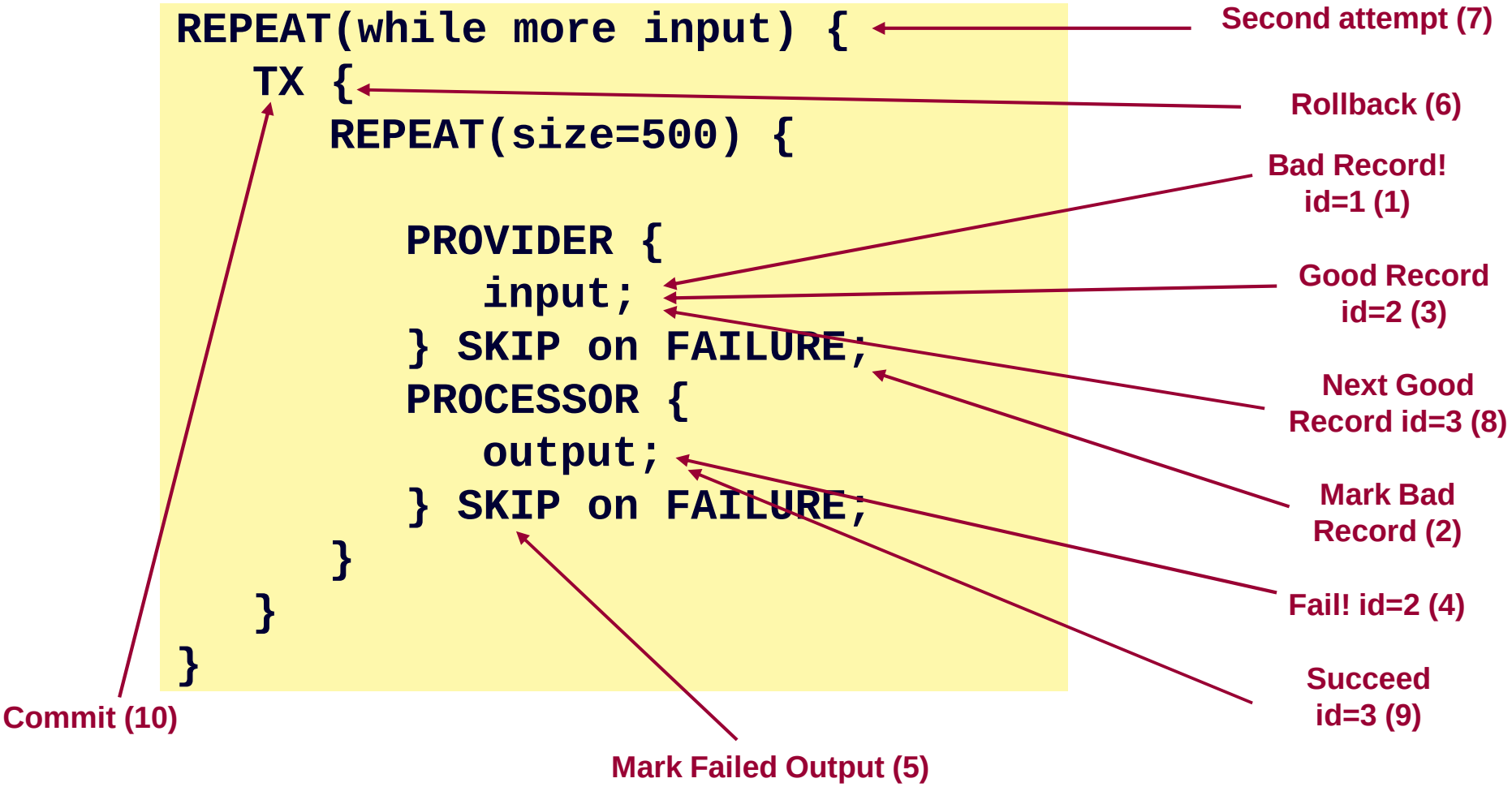
- RepeatTemplate has Exception Policy
  - Do not re-throw “non-critical” exception
  - Re-throw exception and terminate job only when skip limit is exceeded
  - Limit could be absolute or relative (e.g. percentage of bad records)
  - Outer REPEAT block is outside a transaction boundary



Failure Scenario: Skip and Log

- Framework can provide “dumb” recovery
- Always log an error, input or output
- Input Exception
  - Validation error
  - Deterministic: will always happen for this record
  - No rollback necessary
- Output Exception
  - Force a rollback
  - Assume for now deterministic
- In both cases Spring Batch marks the input source to skip the bad record if it is encountered again

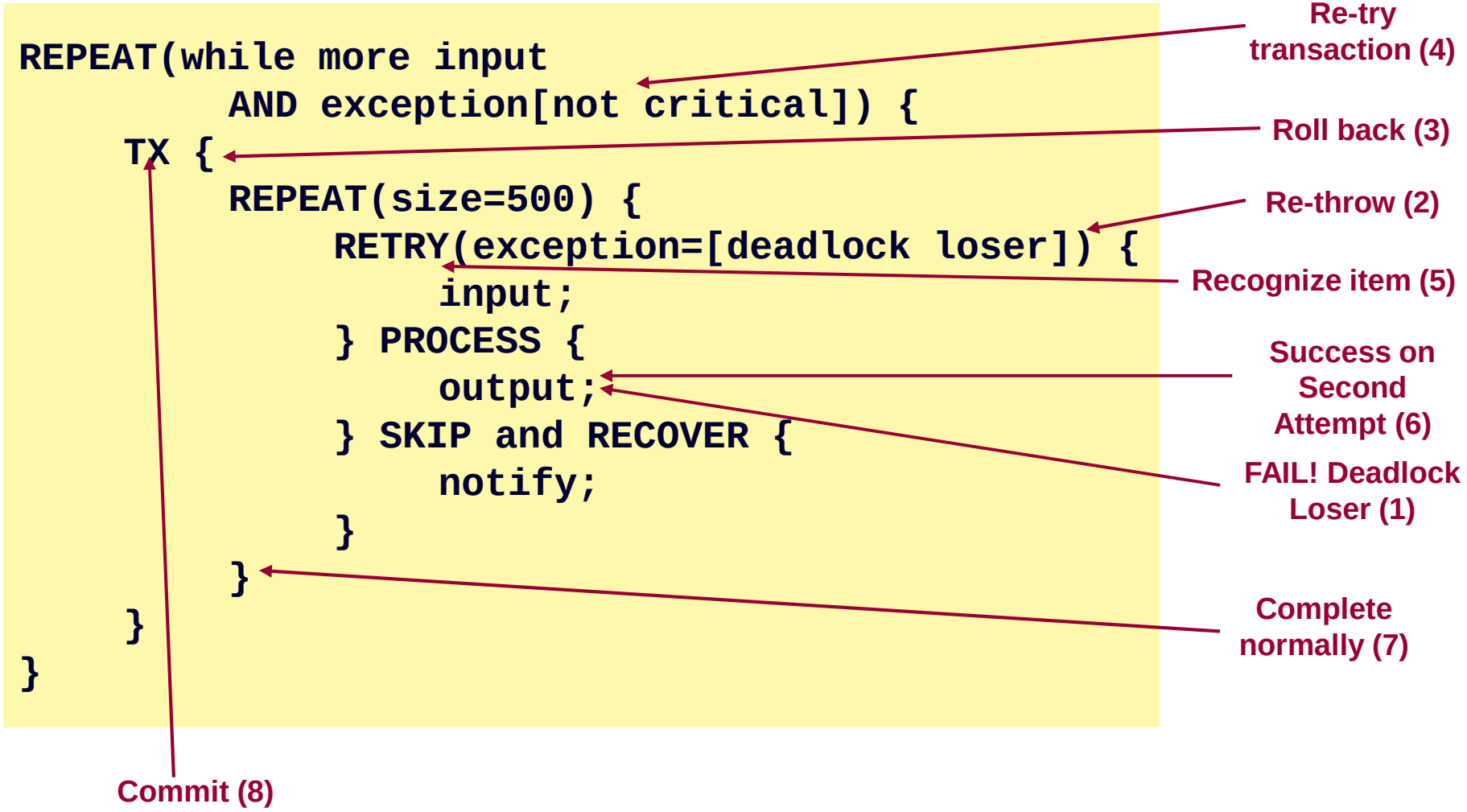
Batch Pseudo Code with Skip



Extended Failure Scenarios: Retry and Recovery

- Inner REPEAT has nested RETRY block
  - Not all output processing exceptions are fatal, e.g. DeadlockLoserException
  - Always re-throw exception – force rollback
  - Stateful – needs to remember identity of failed items
  - Take recovery path when all attempts exhausted
  - Inside a transaction boundary

Successful Retry



Retry with Recovery

```
REPEAT(while more input
        AND exception[not critical]) {
   TX {
     REPEAT(size=500) {
        RETRY(exception=[deadlock loser]) {
           input;
        } PROCESS {
           output;
        } SKIP and RECOVER {
           notify;
        }
     }
   }
}
```

Re-try transaction (4)

Roll back (3)

Re-throw (2)

Recognize item (5)

FAIL! Primary Key Violation (1)

Recover (6)

Complete normally (7)

Commit (8)

Applying the Spring Programming Model to Batch Processing

- Write simple POJO application code encapsulating business logic
  - ItemProvider – iterator-style interface, provides a new record / data item to process
  - ItemProcessor – processes the item
- Add batch behaviour and optimisations non-invasively
  - RepeatTemplate – repeat a block of business processing, terminating according to a policy
  - RetryTemplate – automatically retry a block in the business layer
- Defer architectural choice
  - No need to decide on scalability requirements when business logic is implemented

Spring Batch:
Annotation Driven Batch Configuration

```
@Scope(BatchScope.STEP)
public class MyProcessorImpl implements MyProcessor {

    @Repeat(commitInterval=5)
    @Process(name="tradeSettlement")
    @Transactional(propagation=NESTED)
    public void settle(Trade trade) throws Exception {

        // do some business processing

    }

}
```

Batch Execution Environment

Infrastructure

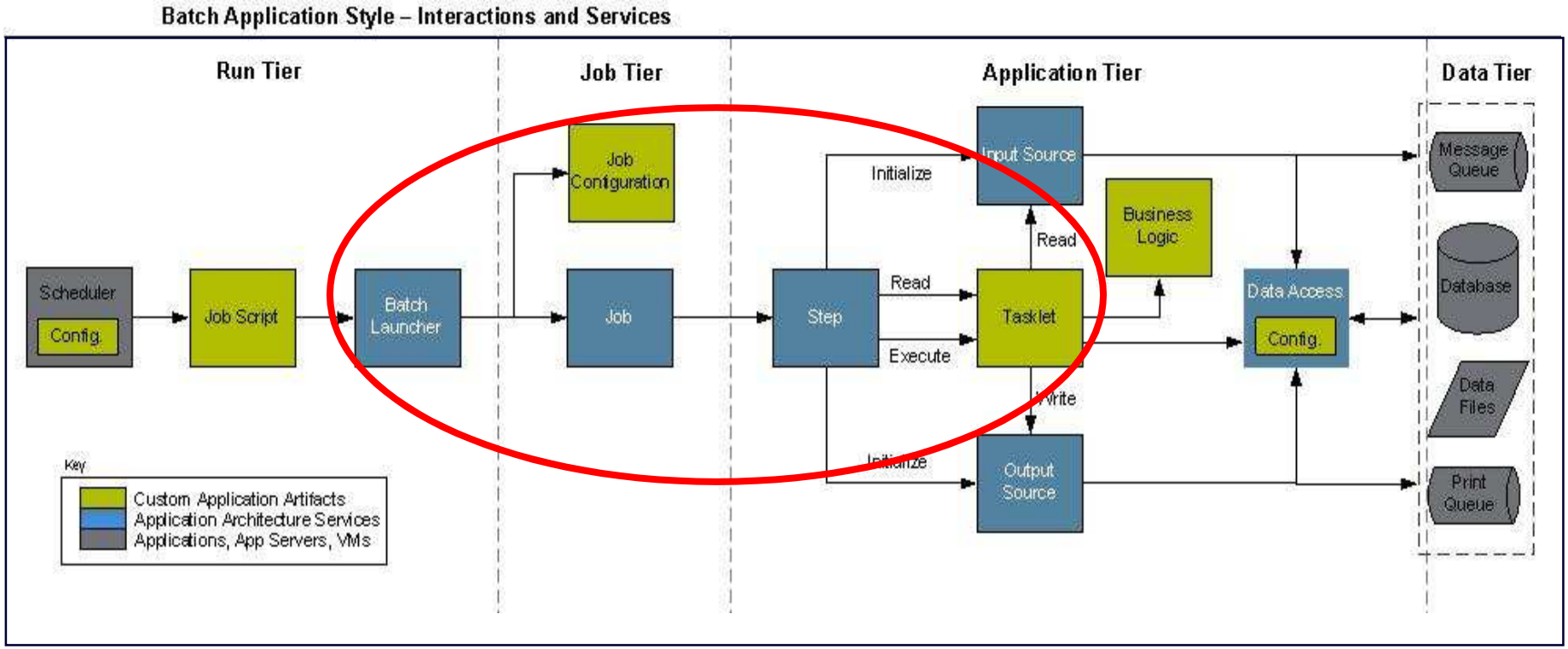
Batch Execution Env

Spring Framework

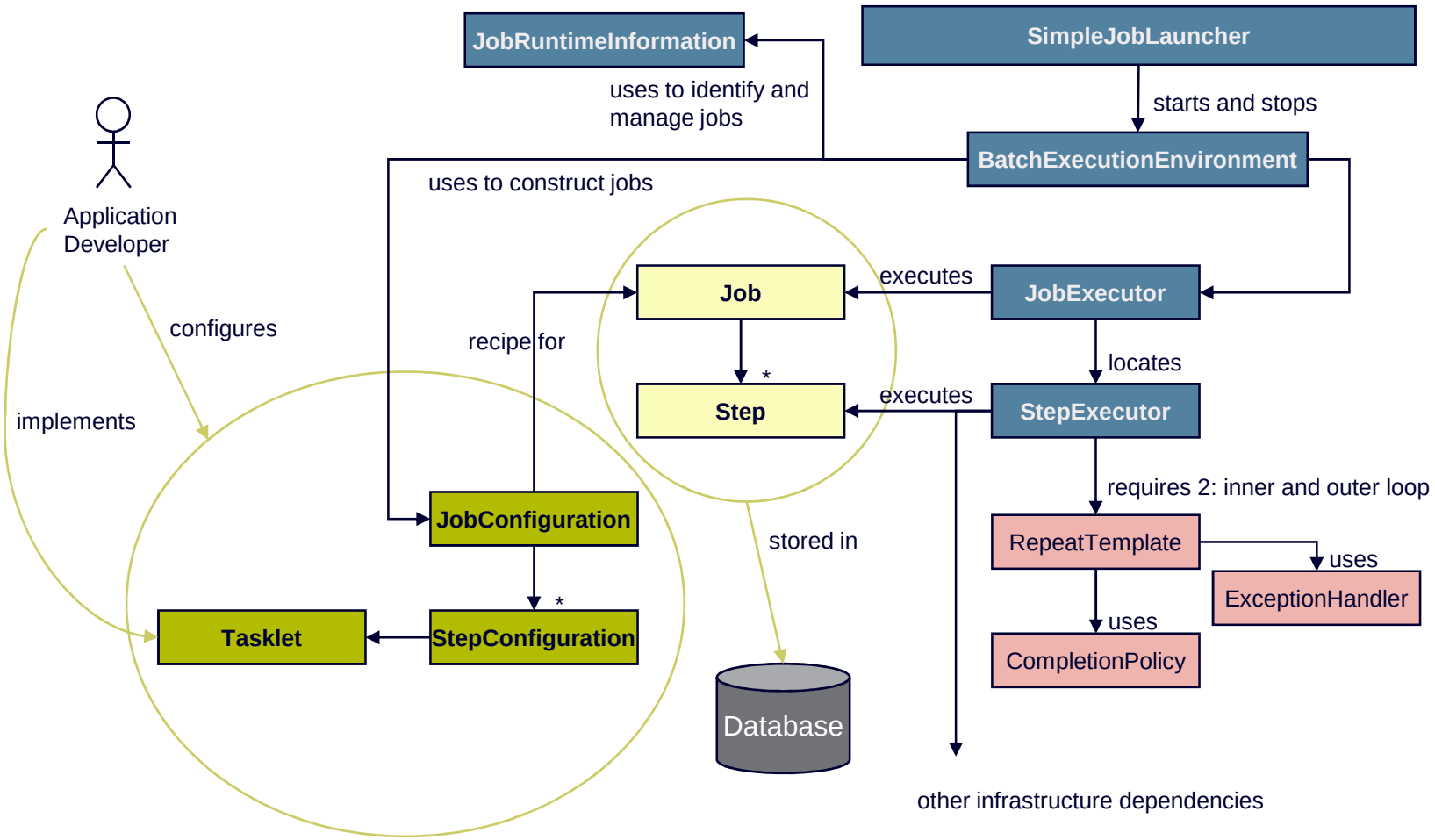
Agenda

- Java Batch – A Missing Enterprise Capability
- Objectives, Scenarios, and Features
- Spring Batch Overview
- Pseudo Code Scenarios
- Batch Domain Reference Model
- Demo
- Roadmap and Summary
- Q & A

Batch Domain Reference Model



Batch Execution Environment Domain Diagram



Simple Batch Pseudo Code

Infrastructure

```
REPEAT(while more input) {

      TX {
         REPEAT(size=500) {
            input;
            output;
         }
      }

}
```

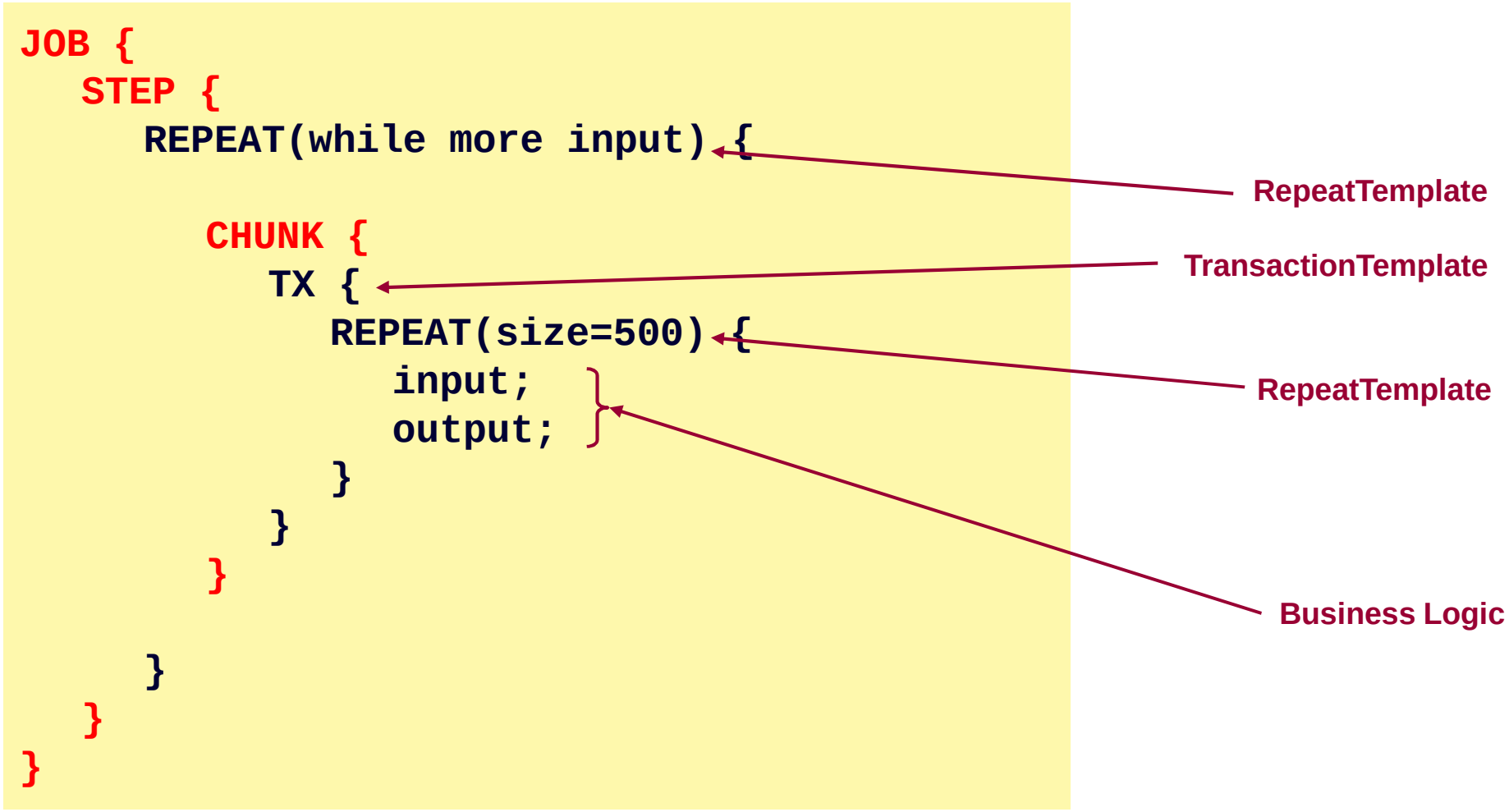
RepeatTemplate

TransactionTemplate

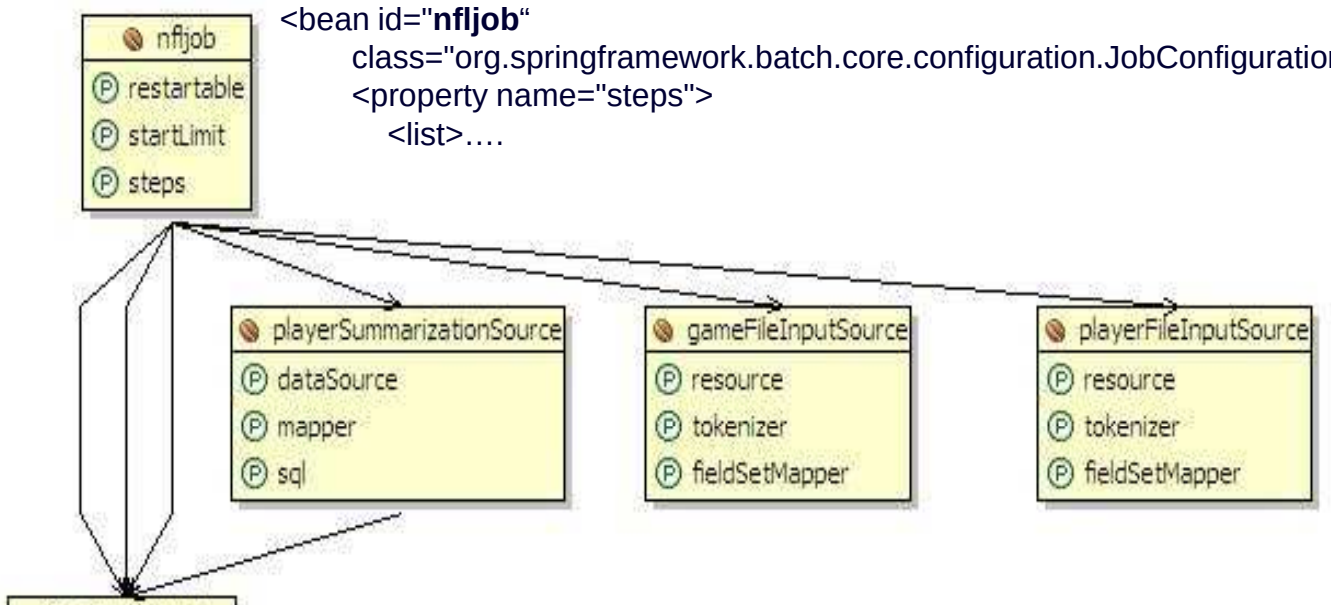
RepeatTemplate

Business Logic

Execution Environment Domain Language



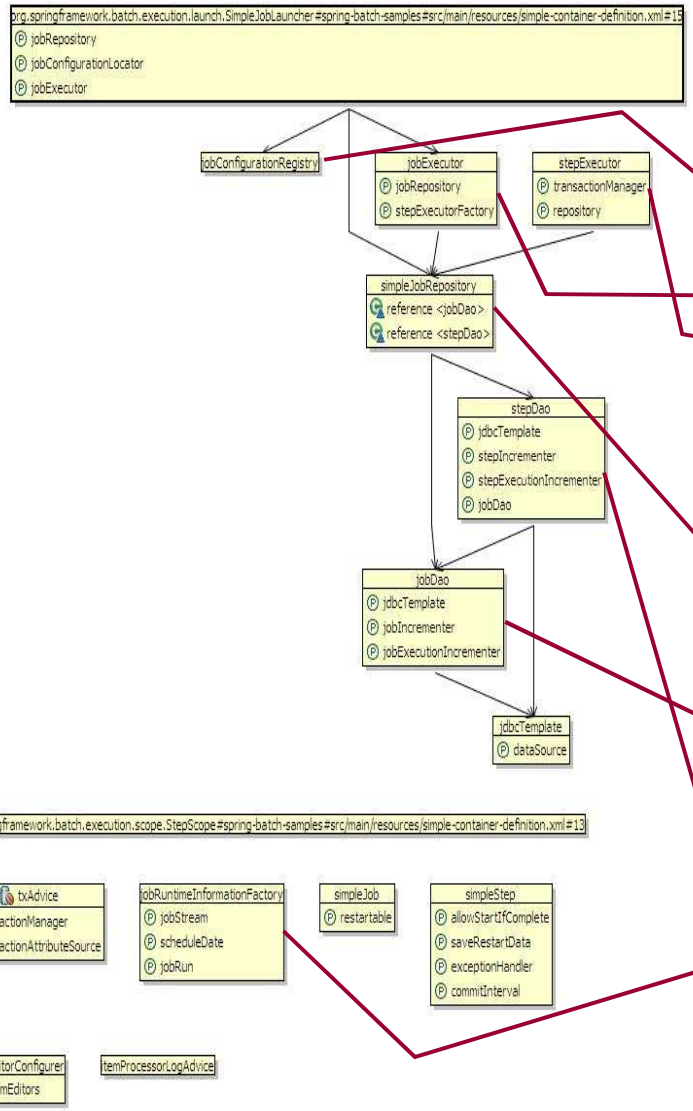
Job Configuration – nfljob sample




```
<bean id="nfljob“
      class="org.springframework.batch.core.configuration.JobConfiguration">
      <property name="steps">
         <list>….
```

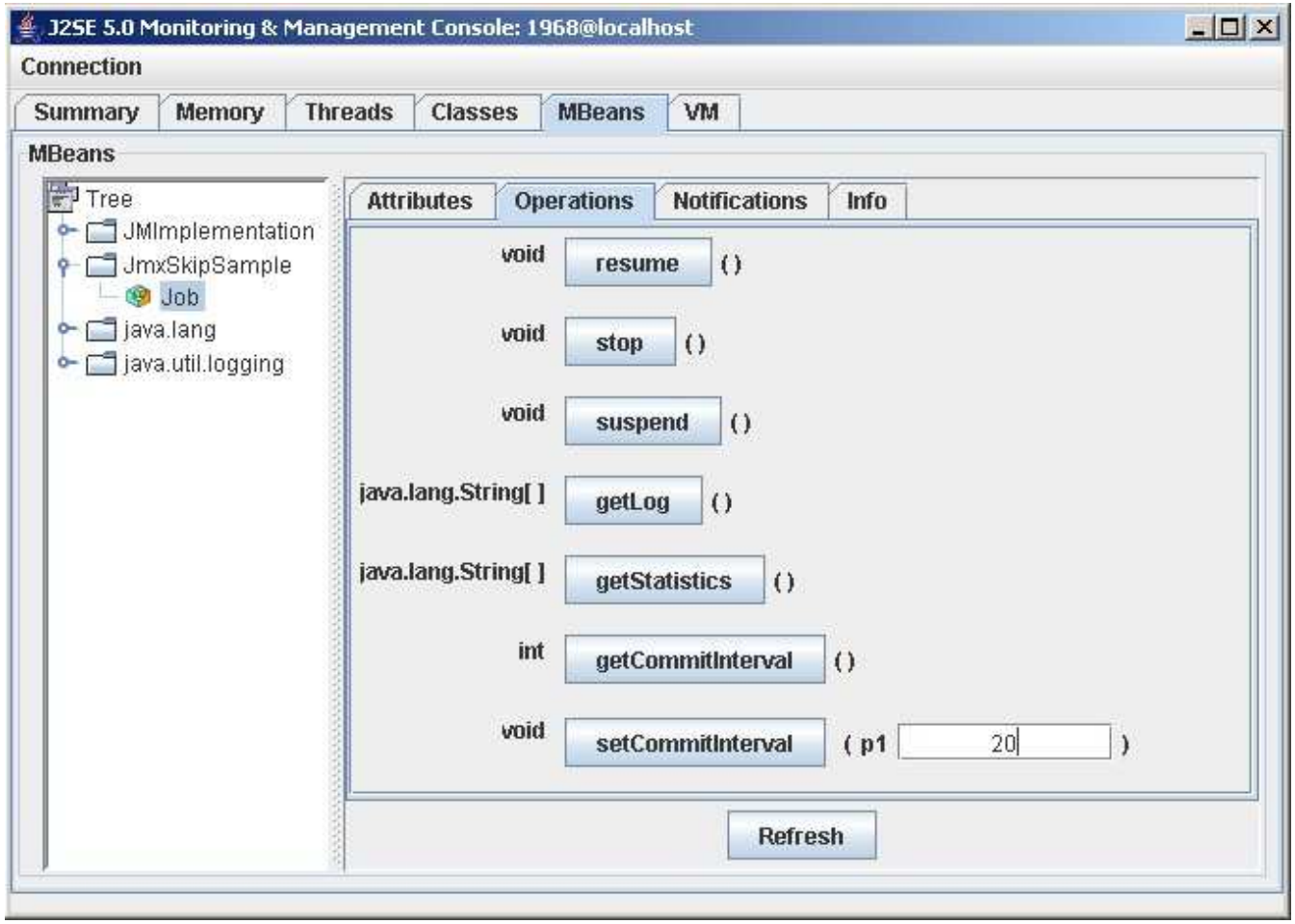
```
<bean id="playerload“
      class="org.springframework.batch.execution.step.SimpleStepConfiguration">
      <property name="commitInterval" value="100"></property>
      <property name="startLimit" value="100" />
      <property name="saveRestartData" value="true" />
      <property name="allowStartIfComplete" value="false" />
      <property name="tasklet">
      <bean class="org.springframework.batch.execution.tasklet.RestartableItemProviderTasklet">
         <property name="itemProvider">
            <bean class="org.springframework.batch.item.provider.InputSourceItemProvider">
               <property name="inputSource“ ref="playerFileInputSource" /> </bean>
```

Batch Execution Environment Configuration

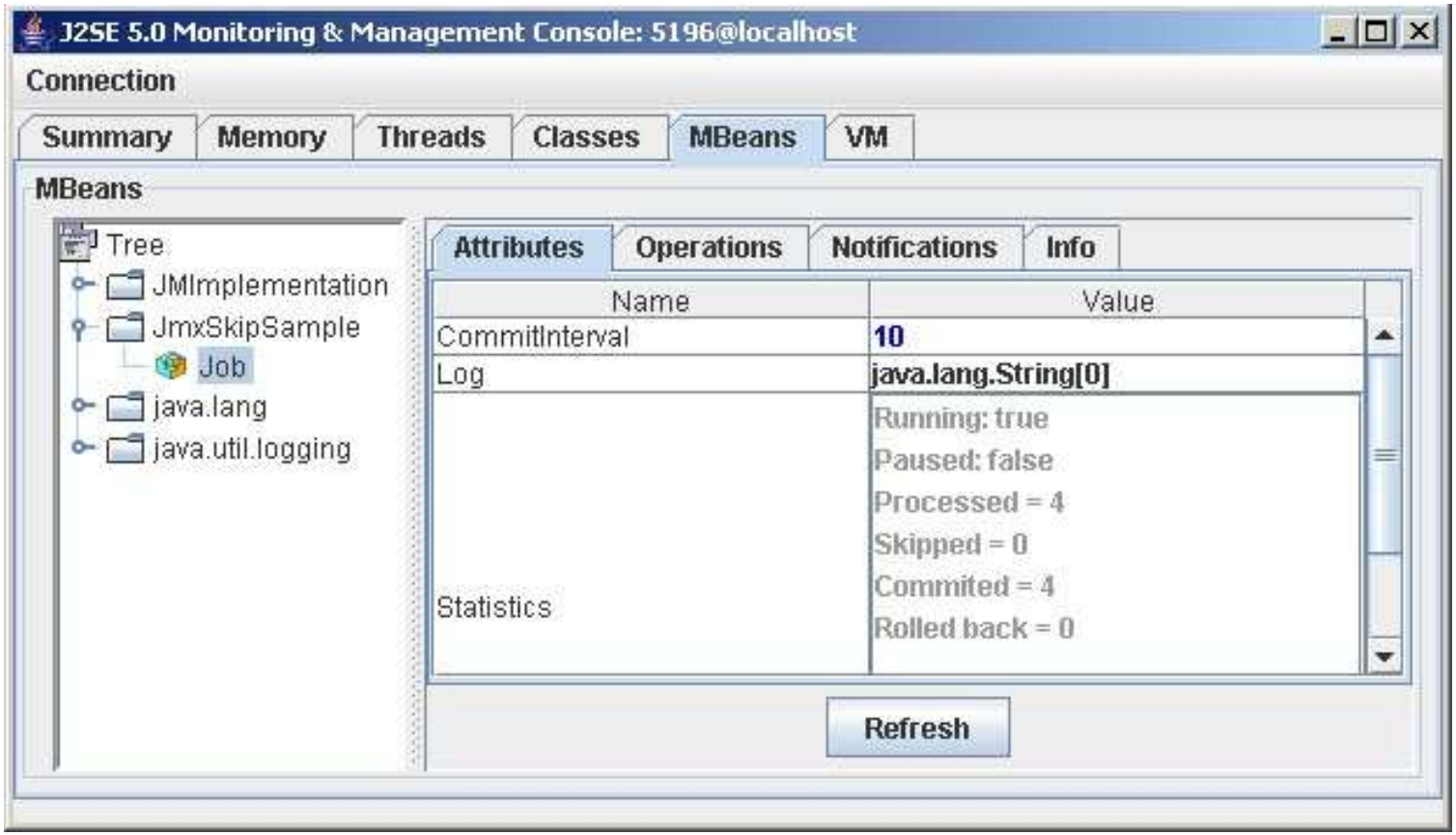



```
<bean class="org.springframework.batch.execution.launch.SimpleJobLauncher">
<property name="jobRepository" ref="simpleJobRepository" />
<property name="jobConfigurationLocator" ref="jobConfigurationRegistry"/>
<property name="jobExecutor" ref="jobExecutor" />
</bean>

<bean id="jobConfigurationRegistry"
class="org.springframework.batch.execution.configuration.MapJobConfigurationRegistry"/>

<bean id="jobExecutor" class="org.springframework.batch.execution.job.DefaultJobExecutor">
<property name="jobRepository" ref="simpleJobRepository" />
<property name="stepExecutorFactory">
<bean class="org.springframework.batch.execution.step.PrototypeBeanStepExecutorFactory">
   <property name="stepExecutorName" value="stepExecutor" />
</bean>
</property>
</bean>

<bean id="simpleJobRepository"
class="org.springframework.batch.execution.repository.SimpleJobRepository">
<constructor-arg ref="jobDao" />
<constructor-arg ref="stepDao" />
</bean>

<bean id="jobDao" class="org.springframework.batch.execution.repository.dao.SqlJobDao">
<property name="jdbcTemplate" ref="jdbcTemplate" />
<property name="jobIncrementer" ref="jobIncrementer" />
<property name="jobExecutionIncrementer" ref="jobExecutionIncrementer" />
</bean>

<bean id="stepDao" class="org.springframework.batch.execution.repository.dao.SqlStepDao">
…. </bean>

<bean id="jobRuntimeInformationFactory“
class="org.springframework.batch.execution.runtime.ScheduledJobIdentifierFactory">
<property name="jobStream" value="TestStream" />
<property name="scheduleDate" value="20070505" />
<property name="jobRun" value="1" />
</bean>
```

Spring Batch:
JMX Management and Operations



Spring Batch:
JMX Management and Operations



Agenda

- Java Batch – A Missing Enterprise Capability
- Objectives, Scenarios, and Features
- Spring Batch Overview
- Pseudo Code Scenarios
- Batch Domain Reference Model
- Demo
- Roadmap and Summary
- Q & A

Spring Batch—Road Map

- Release 1.0
  - Infrastructure support of the development of batch execution environments
    - Repeat—batch operations together
    - Retry—retry a piece of work if there is an exception
    - File and database I/O templates
  - Simple Batch Execution implementation providing full, robust management of the batch lifecycle
    - Restart, skip, statistics, status
    - Validation/record processing
    - Batch monitoring and management (coming soon)
- 1.0-m2 is in use at roughly 40+ projects with at least 3 in production
- Future (1.1) might include…
    - Partitioned Batch Execution Environment, SEDA, Grid, ESB support, and more…

Sample Jobs

Job/Feature	delimited input	fixed-length input	xml input	multiline input	db driving query input	db curosr input	delimited output	fixed length output	xml output	multiline output	db output	skip	restart	quartz
fixedLengthImportJob		X									X			
multilineJob		X		X										
multi lineOrderJob	X			X				X		X				
quartzBatch														X
simpleModuleJob		X									X			
simpleSkipSample												X		
skipWithRestartSample												X	X	
sqlCursorTradeJob						X								
tradeJob	X				X		X							
xmlJob			X						X					

Spring Batch Team

- **Accenture**
  - Scott Wintermute
  - Wayne Lund
  - Lucas Ward
  - Waseem Malik
  - Amir Haq
  - Paulo Villela
  - Peter Zozom
  - Tomas Slanina
  - Robert Kasanicky
  - Tomi Vanek
- **JReflections**
  - Kerry O'Brien
- **Interface21**
  - Dave Syer
  - Rob Harrop
  - Ben Hale
  - Christian Dupuis

Spring Batch Resources

- Spring Batch Source Code: https://springframework.svn.sourceforge.net/svnroot/springframework/spring-batch/trunk/
- Forum: http://forum.springframework.org/forumdisplay.php?f=41
- Mailing List: http://lists.interface21.com/listmanager/listinfo/spring-batch-announce
- Issue Tracking (Jira) http://opensource.atlassian.com/projects/spring/browse/BATCH
- CI Server (Bamboo) http://build.springframework.org:8085/bamboo/browse/BATCH

Summary

- Lack of a standard enterprise batch architecture is resulting in higher costs associated with the quality and delivery of solutions.
- Spring Batch provides a highly scalable, easy-to-use, customizable, industry-accepted batch framework collaboratively developed by Accenture and Interface21
- Spring patterns and practices have been leveraged allowing developers to focus on business logic, while enterprise architects can customize and extend architecture concerns

accenture

High performance. Delivered.

Q&A