Taking Java™ Technology to New Frontiers: Enterprise Batch Processing With Spring Batch

Rod Johnson—Interface21

Scott Wintermute—Accenture

Wayne Lund—Accenture

TS-76950

Goal of This Talk

What You Will Gain

How to implement reliable enterprise batch-processing with Spring Batch.

Agenda

Java Batch Technology:
A Missing Enterprise Capability

Spring Batch:
An Accenture and Interface21 Partnership

Objectives, Scenarios, and Features

Spring Batch Overview

Pseudo Code Scenarios

Demo

Roadmap and Summary

Q&A

Java Batch Technology — A Missing Enterprise Capability in the Market

- Automates complex processing of large volumes of data and/or transactions most efficiently processed without user interaction
- Batch jobs are part of most IT projects
  - Currently no commercial or open source framework provides a robust, enterprise-scale solution/framework
- Lack of a standard architecture has resulted in the proliferation of expensive one-off, in-house custom architectures
- Batch processing is used to process **billions** of transactions everyday within mission-critical enterprise applications

Agenda

Java Batch Technology:
A Missing Enterprise Capability

Spring Batch:
An Accenture and Interface21 Partnership

Objectives, Scenarios, and Features

Spring Batch Overview

Pseudo Code Scenarios

Demo

Roadmap and Summary

Q&A

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 Technology:
- A Missing Enterprise Capability

Spring Batch:
- An Accenture and Interface21 Partnership

Objectives, Scenarios, and Features

Spring Batch Overview

Pseudo Code Scenarios

Demo

Roadmap and Summary

Q&A

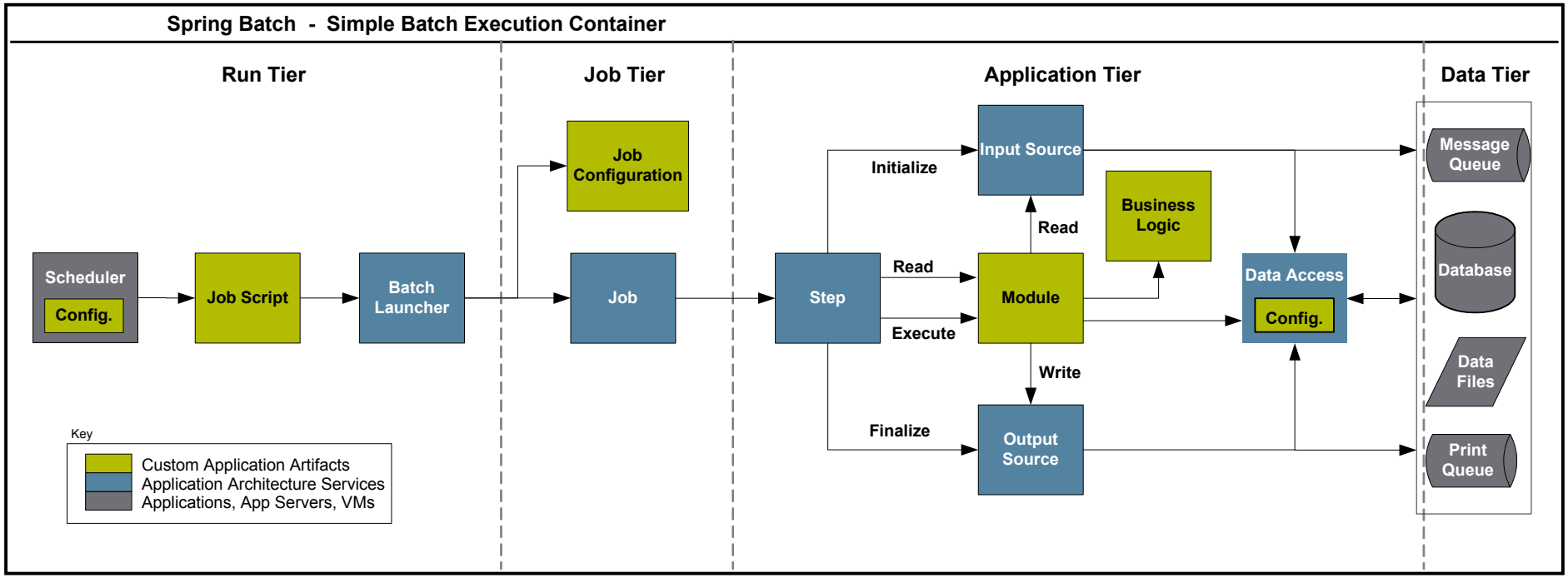
Spring Batch: Technical and Architecture Objectives

- Clear separation of concerns between the infrastructure, batch execution container, and the batch application
- Provide common, batch container services as interfaces
- Provide simple and default implementations of the batch container interfaces
- Easy to configure, customize, and extend services, by leveraging the Spring framework across all layers
- Batch container services should be easy to replace or extend, without impact to the infrastructure layer
- Provide a simple deployment model built using Maven

Spring Batch: Business Scenarios

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

Batch Reference Model



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 Technology:
- A Missing Enterprise Capability

Spring Batch:
- An Accenture and Interface21 Partnership

Objectives, Scenarios, and Features

Spring Batch Overview

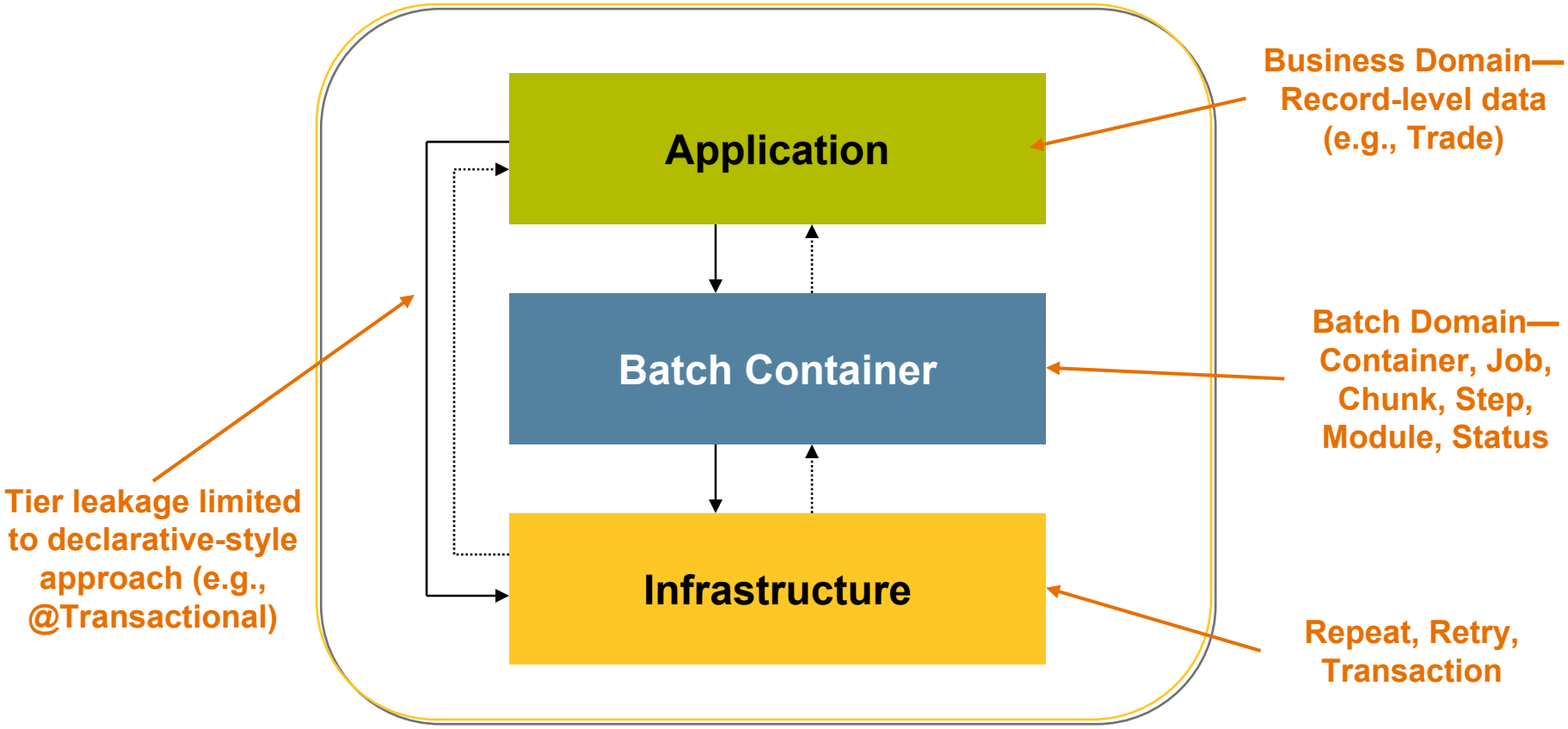
Pseudo Code Scenarios

Demo

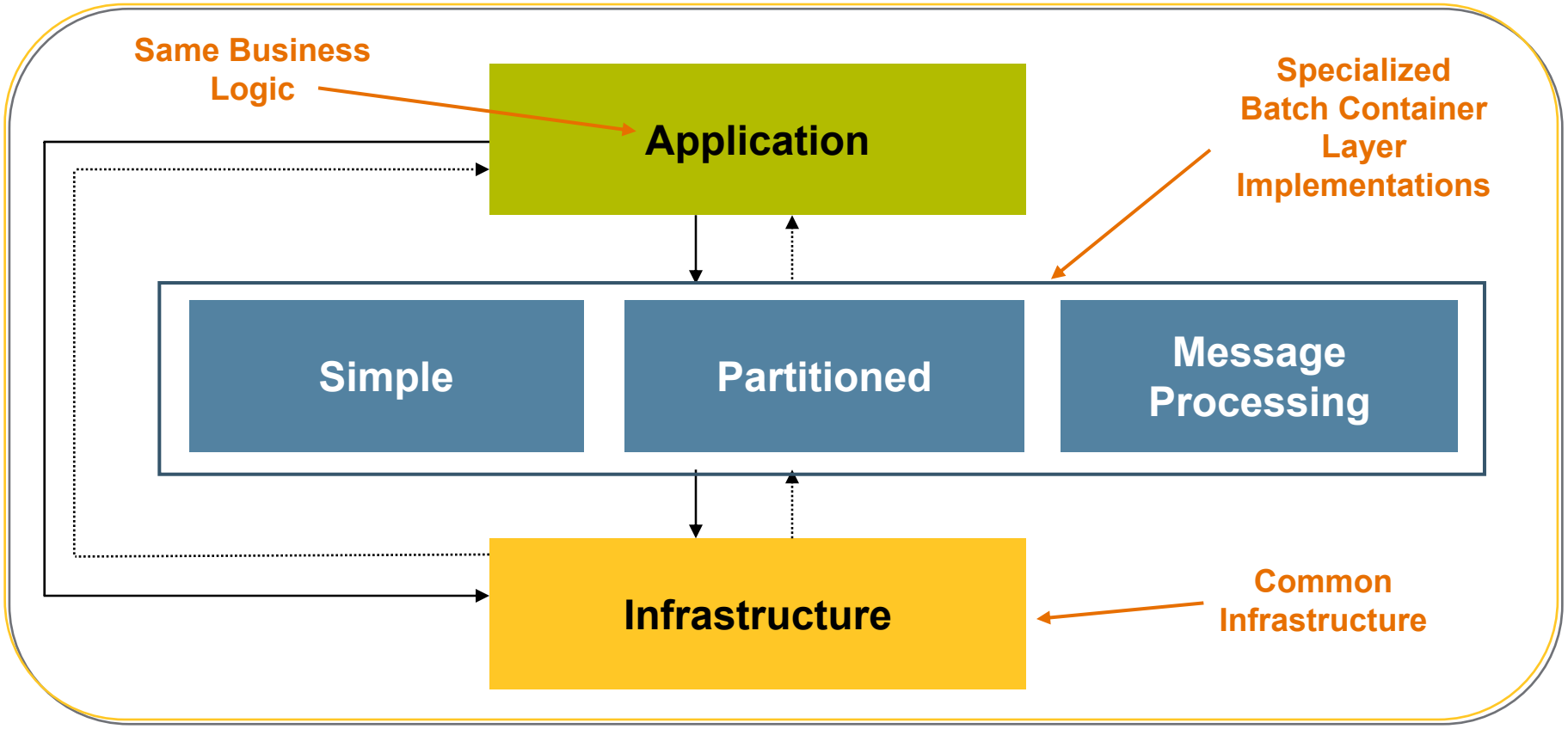
Roadmap and Summary

Q&A

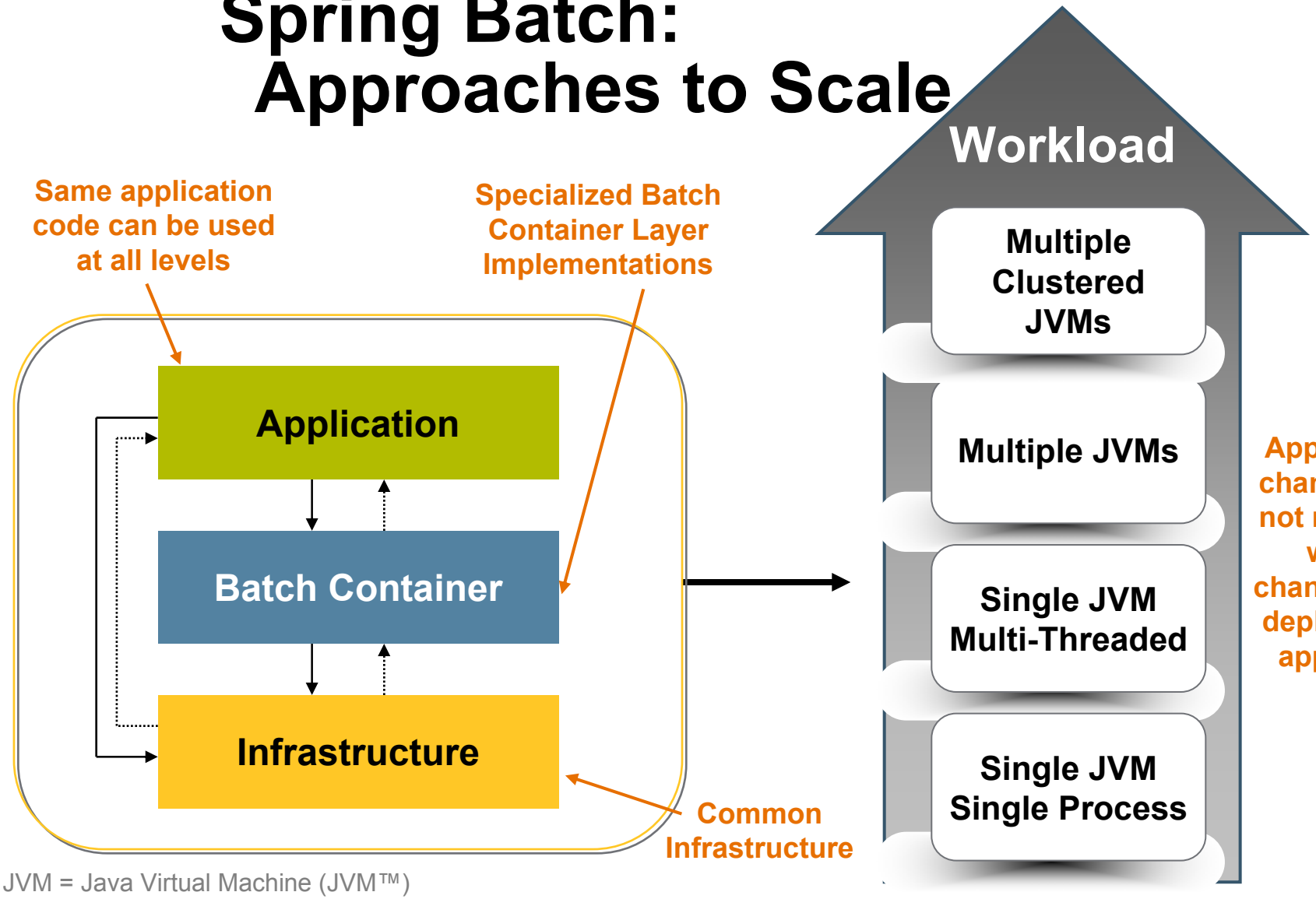
Spring Batch: Layered Architecture



Spring Batch: Container Implementations



Spring Batch: Approaches to Scale



JVM = Java Virtual Machine (JVM™)
The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java™ platform.

Agenda

Java Batch Technology:
- A Missing Enterprise Capability

Spring Batch:
- An Accenture and Interface21 Partnership

Objectives, Scenarios, and Features

Spring Batch Overview

Pseudo Code Scenarios

Demo

Roadmap and Summary

Q&A

Spring Batch: Infrastructure

- All the business scenarios can be implemented in terms of lower level concepts
- Identify three such infrastructure concepts
  - **Transaction**—atomic, consistent, isolated, durable
  - **Repeat**—continually execute an operation **until** it completes successfully or **unless** it fails
  - **Retry**—automatically retry a failed operation **until** it succeeds or **until** a recovery action is taken
- All of the above can be nested and composed together to build “container” features

Simple Batch Pseudo Code

```
JOB processJob: {
        STEP processStep: {
                ITERATE(Until Module Complete){
                        TX{
                                ITERATE(Until CommitPolicy = true){
                                        Try {
                                                Module process {: object |
                                                        Object o = INPUT { Retu
                                                        OUTPUT {: o }
                                                }
                                        } Catch (INPUT_ERROR){
                                                Log(INPUT_ERROR.INPUT)
                                                continue
                                        } Catch (OUTPUT_ERROR) {
                                                Log(OUTPUT_ERROR.INPUT)
                                                Rollback()
                                        } Catch (CRITICAL_ERROR){
                                                Terminate()
                                        }
                                }
                        }
                }
        }
}
```

BatchTemplate

BatchTemplate

TransactionTemplate

Business Logic

Policy-Driven Pseudo Code

```
JOB processJob: {
            STEP processStep: {
                        ITERATE(Until Module Complete){
                                    TX{
                                                ITERATE(Until CommitPolicy = true){
                                                            Try {
                                                                        Try{
                                                                                    Module process {: objec
                                                                                                Object o = IN
                                                                                                if(RetryPolic
                                                                                                            OUT
                                                                                                else
                                                                                                            REC
                                                                                    }
                                                                        }Catch(ERROR){
                                                                                    RetryPolicy.registerErr
                                                                                    rethrow(ERROR)
                                                                        }
                                                            } Catch (INPUT_ERROR){
                                                                        Log(INPUT_ERROR.INPUT)
                                                                        continue
                                                            } Catch (OUTPUT_ERROR) {
                                                                        Log(OUT_ERROR.INPUT)
                                                                        Rollback()
                                                            } Catch (CRITICAL_ERROR){
                                                                        Terminate()
}           }           }           }           }           }
```

BatchTemplate

BatchTemplate

TransactionTemplate

Business Logic

Input and Output Retries

- Input Retry—Skip
  - Re-throw exception only when no policies are successful
  - Normally outside a transaction boundary
  - Alternatively can be used to retry a non-transactional operation (e.g., call to web service)
- Output Retry—Recover
  - Always re-throw exception—force rollback
  - Take recovery path when all attempts exhausted
  - Normally inside a transaction boundary

Input Retry—Skip

```
ITERATE(Until Module Complete){
        RETRY(chunk) {
                TX      {
                        ITERATE(Until CommitPolicy = true){
                                RETRY(input) {
                                input;
                        } PROCESS {
                                output;
                        } RECOVER {
                                skip;
                        }
                }
                }
        }
}
```

Recover (2)

FAIL! (1)

Skip (3)

Complete normally (4)

Output Retry—Recover

```
ITERATE(Until Module Complete){
        RETRY(chunk)  {
                TX      {
                        ITERATE(Until CommitPolicy = true){
                                RETRY(input)  {
                                input;
                        } PROCESS  {
                                output;
                        } RECOVER  {
                                recover;
                        }
                }
                }
        }
}
```

Re-try transaction (4)

Roll back (3)

Re-throw (2)

Success on Second Attempt (5)

FAIL! (1)

Complete normally (6)

Commit (7)

Applying the Spring Programming Model to Batch Processing

- Write simple POJO application code encapsulating business logic
  - **`DataProvider`**—iterator-style interface, provides a new record/data item to process
  - **`DataProcessor`**—processes the item
- Add batch behaviour declaratively
  - **`BatchTemplate`**—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

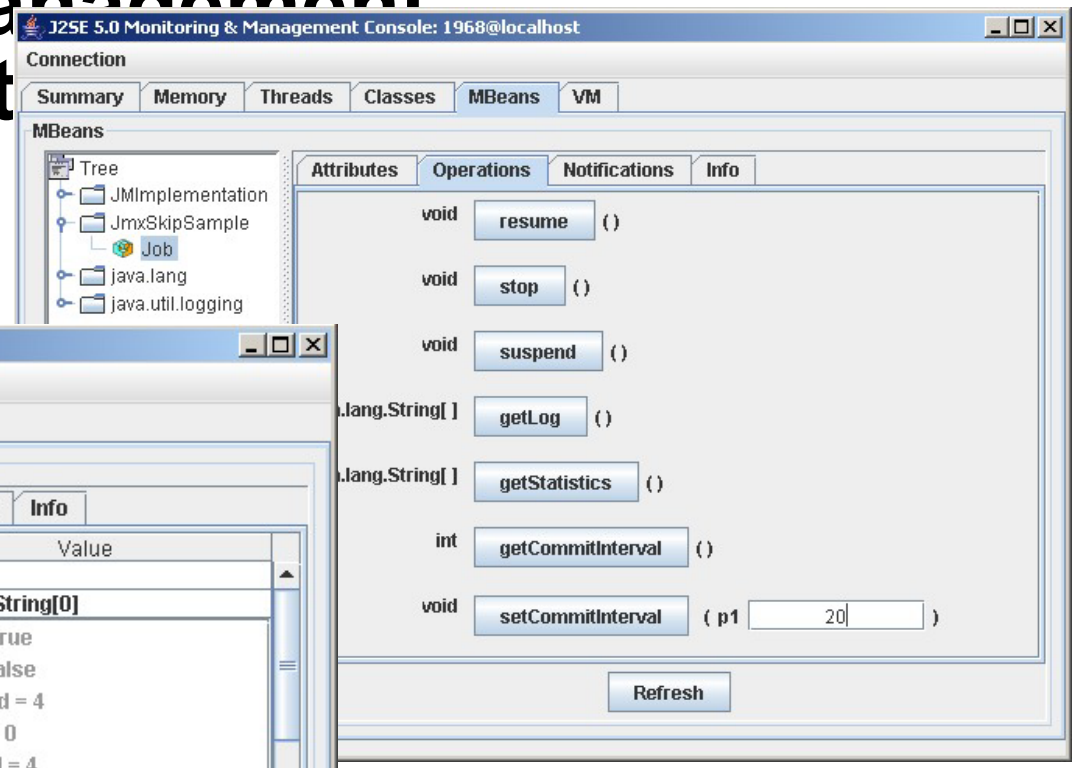
```
public class MyProcessorImpl implements MyProcessor {

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

        // do some business processing

    }

}
```

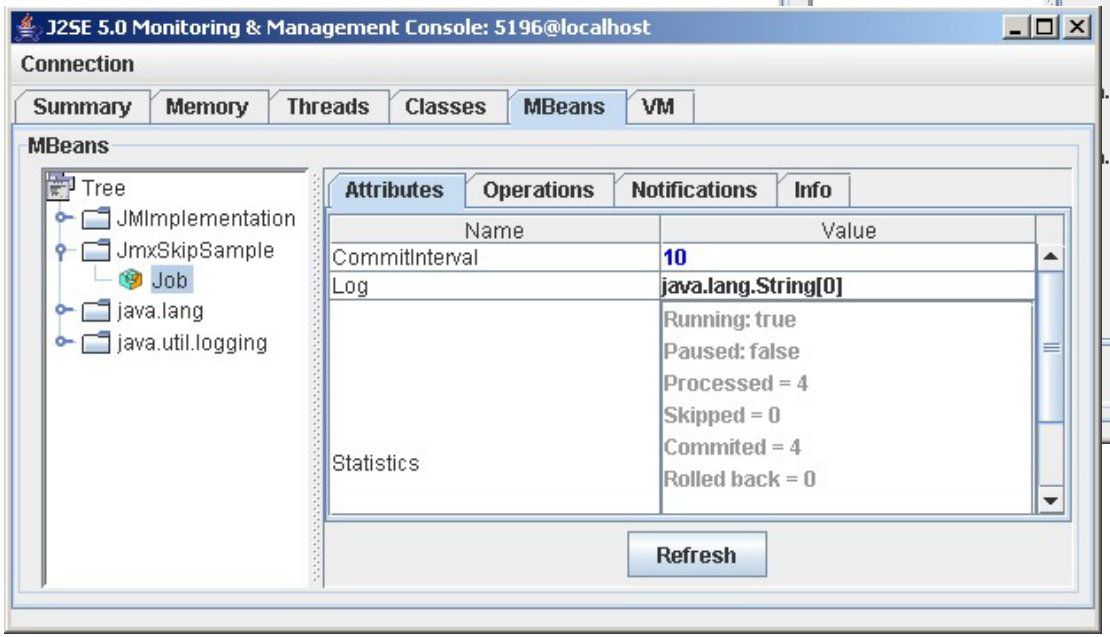
Container

Infrastructure

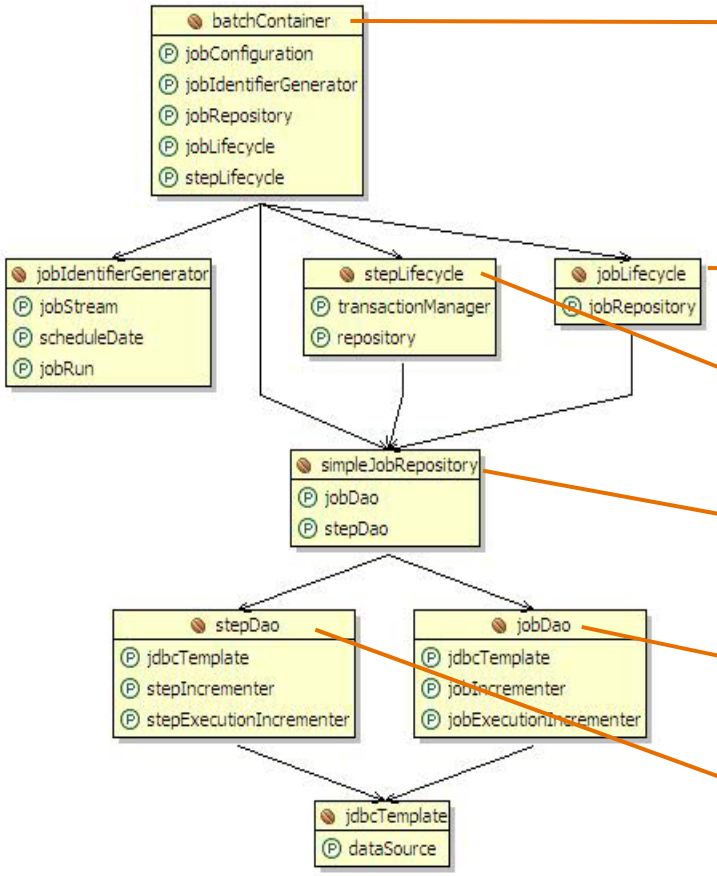
Spring Framework

Spring Batch: Java Management Extensions (JMX™) Management and Operat



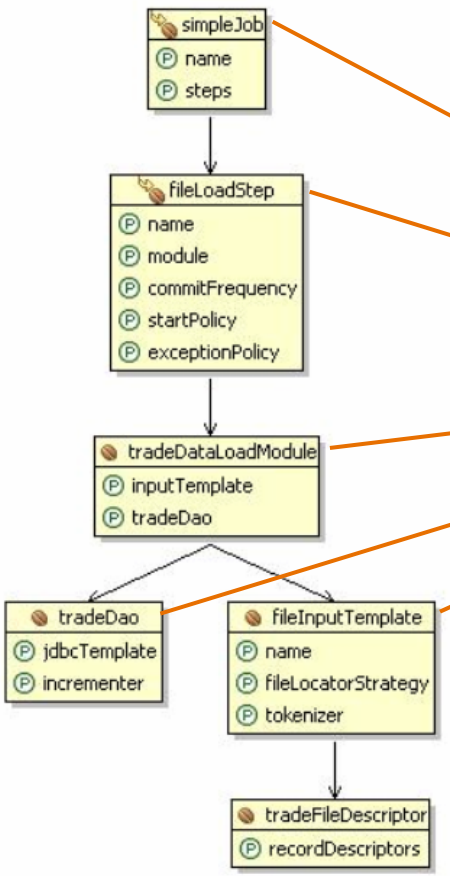


Container Configuration




```
<bean id="batchContainer" class="SimpleBatchContainer" >
<property name="jobConfiguration" ref="jobConfiguration" />
<property name="jobIdentifierGenerator" ref="jobIdentifierGenerator" />
<property name="jobRepository" ref="simpleJobRepository" />
<property name="jobLifecycle" ref="jobLifecycle" />
<property name="stepLifecycle" ref="stepLifecycle" />
</bean>

<bean id="jobLifecycle" class="DefaultJobLifecycle" >
<property name="jobRepository" ref="simpleJobRepository" />
</bean>

<bean id="stepLifecycle“ class="SimpleStepLifecycle">
<property name="repository" ref="simpleJobRepository" />
</bean>

<bean id="simpleJobRepository" class="SimpleJobRepository">
<property name="jobDao" ref="jobDao" />
<property name="stepDao" ref="stepDao" />
</bean>

<bean id="jobDao" class="SqlJobDao" >
<property name="jdbcTemplate" ref="jdbcTemplate" />
…</bean>

<bean id="stepDao“ …. </bean>
<bean id="jobIdentifierGenerator" class="DefaultJobIdentifierGenerator"> …</bean>
…
```

Job Configuration




```
<bean id="simpleModuleJob" parent="Job">
  <property name="name" value="fixedLengthImportJob" />
  <property name="steps">
    <list>
      <bean id="tradeStep" parent="Step">
        <property name="name" value="ImportTradeDataStep" />
        <property name="module">
         <bean class="module.SimpleTradeModule">
           <property name="tradeDbWriter" ref="tradeWriter" />
           <property name="inputTemplate" ref="fileInputTemplate" />
         </bean>
        </property>
        <property name="commitInterval" value="5" />
      </bean>
    </list>
  </property>
</bean>
```

DEMO

Agenda

Java Batch Technology:
- A Missing Enterprise Capability

Spring Batch:
- An Accenture and Interface21 Partnership

Objectives, Scenarios, and Features

Spring Batch Overview

Pseudo Code Scenarios

Demo

Roadmap and Summary

Q&A

Spring Batch—Road Map

- Release 1.0M2 (tbd)
  - Infrastructure support of the development of containers
    - Repeat—batch operations together
    - Retry—retry a piece of work if there is an exception
    - File and database I/O templates
  - Simple Batch Execution Container providing full, robust management of the batch lifecycle
    - Restart, skip, statistics, status
    - Validation/record processing
    - Batch monitoring and management
- Future might include…
    - Partitioned container, SEDA, Grid, ESB support, and more…

Reference Applications

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										
multielineOrderJob	X			X				X		X				
quartzBatch														X
simpleModuleJob		X									X			
simpleSkipSample												X		
skipWithRestartSample												X	X	
sqlCursorTradeJob						X								
tradeJob	X				X		X							
xmlJob			X						X					

Agenda

Java Batch Technology:
- A Missing Enterprise Capability

Spring Batch:
- An Accenture and Interface21 Partnership

Objectives, Scenarios, and Features

Spring Batch Overview

Pseudo Code Scenarios

Demo

Roadmap and Summary

Q&A

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

Q&A

-
-
-

Taking Java™ Technology to New Frontiers: Enterprise Batch Processing With Spring Batch

Rod Johnson—Interface21

Scott Wintermute—Accenture

Wayne Lund—Accenture

TS-76950