Open Source in Public Sector

Holger Zobel

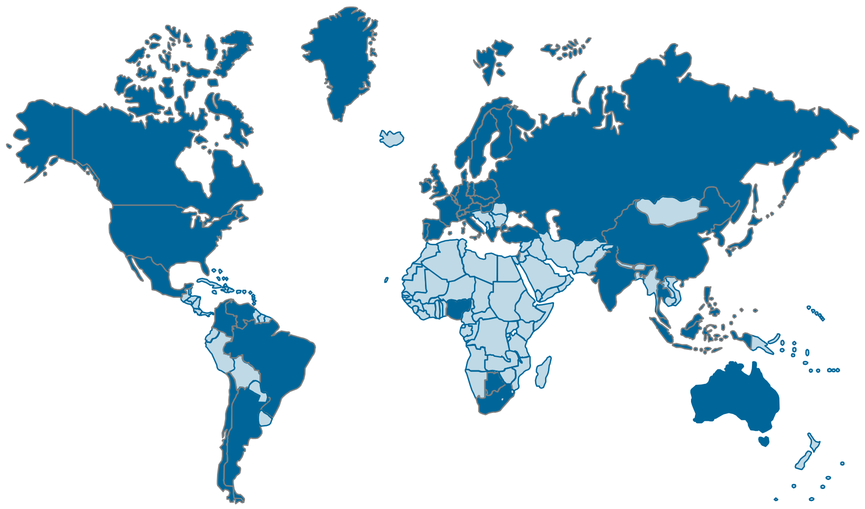
2008-04-21

Accenture

Management Consulting

System Integration & Technology

Outsourcing



🟦 Countries with one or more Accenture offices

Agenda

1. **Public sector and Open Source**
2. Case study: Pension Project
3. Spring Batch

Public Sector Policies

- The ministry responsible for ICT is “Fornyingsdepartementet”

- Key policies:
  - eNorway 2009
  - Mandatory Open Standards
  - Currently strong support of Open Source by the Minister.

eNorway 2009

- eNorway 2009 has three target areas:
  - The individual in the digital Norway
  - Innovation and growth in business and industry
  - A coordinated and user-adapted public sector

Use of Open Standards and open Source Applications

- “Initiatives should be undertaken to develop open source skills in the public sector. A recommendation concerning use of open source software in the public sector will be developed in 2005. By the end of 2006, all public sector agencies shall have drawn up plans for use of open source applications.”

- Goals
  - By 2009, all new ICT and information systems in the public sector shall use open standards.
  - By 2006, a set of administration standards for data and document exchange shall have been established.
  - By 2006, all public sector agencies shall have incorporated how they are going to use open standards, service-oriented architecture and open source applications in the relevant planning documents.
  - By 2008, data and document exchange in the public sector shall satisfy administration standards.
  - By 2008, all official forms shall be available electronically and built round a common user interface.

Mandatory Open Standards

- HTML - 4.01/ XHTML - 1.0 is recommended for public sector web sites
- In special circumstances, like need to keep formatting or documents for further editing, the following formats shall be used:
  - PDF 1.4 or PDF/A - ISO 19005-1 is mandatory for documents which should not be edited
  - ODF 1.0 (Second Edition), ISO/IEC 26300:2006 is mandatory, but it is suggested that other formats be used in parallel.
  - OOXML is observed and will be evaluated later

Licences

- Public sector seldom has constraints on keeping trade secrets
- But still have to consider licences to make sure that they are compatible.
  - Example: Using proprietary source code and GPL in the same project.

Open sourcing software owned by Public Sector

- Almost no software is open sourced by public sector organisations.
- Reasons include:
  - No culture for sharing
  - Few people with experience with open source
  - Very specialised applications like Child Support administration
  - Security

Agenda

1. Public sector and Open Source
2. **Case study: Pension Project**
3. Spring Batch

The Pension Reform - One of the largest reforms in public sector

- The Norwegian Labour and Welfare Organisation (NAV) is responsible for government pensions, unemployment benefits, children's allowance, disability and several other government benefits. NAV has 14 000 employees and is administrating one third of the Norwegian governments budget.
- The Norwegian Parliament decided on the Pension reform in the spring session of 2005. The reform is planned to be effective as of 2010.
- There is broad political agreement upon the principles in the reform
- A Government White Paper about Old Age Pensions in the National Insurance was released November 2006

Opptjening og uttak av alderspensjon i folketrygden

St.meld. nr. 5 (2006 – 2007)

20. oktober 2006

The new Pension Application

The pension project is developing two front end applications:

- Internet self service application
- Internal Case Worker Application (2000 users)

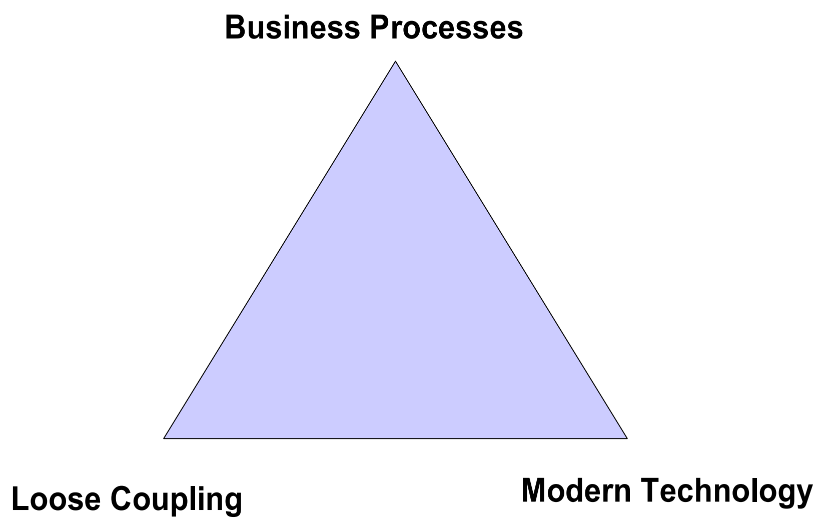
The project also develops six new backend systems and integrates with 14 existing backend systems based on different platforms and technologies.

The first release of the application was in December 2007.

Technical Architecture Vision and Strategy

Technical Architecture Vision:

- The Pension Reform Project's platform will establish the foundation for The Client's preferred future technical platform.
- The solution implemented will be available to anyone, anywhere, anytime.
- The platform is modern in 2010, allowing room for changes in technology and new functionality within a reasonable timeframe, as seen from the system management perspective.



Architecture Strategy:

- Service Oriented Architecture
- Layering
- End to End Operation
- Virtualisation

Technical Architecture

- Java based custom application
  - Documentation
  - Example Code
- SOA Platform
  - Service Identification
  - Guidelines and connectivity
  - Developer Training and Certification
- Security
  - Identity and Access Management (I & AM)
- Rule engine

Selection Criteria

- Functionality
- Standards based
- Future proof
- Support and available competent people
- Price

Technical Platform

- Red Had Linux
- Java Frameworks (next slide)
- WebSphere Application Server
- WebSphere Process Server
- Rational Application Developer (Eclipse based)
- Tivoli Access Manager / Tivoli Identity Manager
- Blaze Advisor (rule engine)
- Content Manager (document archive)
- VMWare for development and test environments

Java Frameworks

- Java EE 5
- Spring
- Apache Commons
- JSF (myfaces)
- Spring Web Flow
- Hibernate
- Dozer
- Ajax4Jsf

Agenda

1. Public sector and Open Source
2. Case study: Pension Project
3. **Spring Batch**

Why Batch?

- Large volumes
- Import/export files
- Convert data
- Archiving

- Not a part of Java EE today

Alternatives for Java developers today

- Do it in Java:
  - Develop an in house architecture
  - Implement business logic in a language you know

- Use an ETL tool
  - Expensive
  - Implement business logic in another language

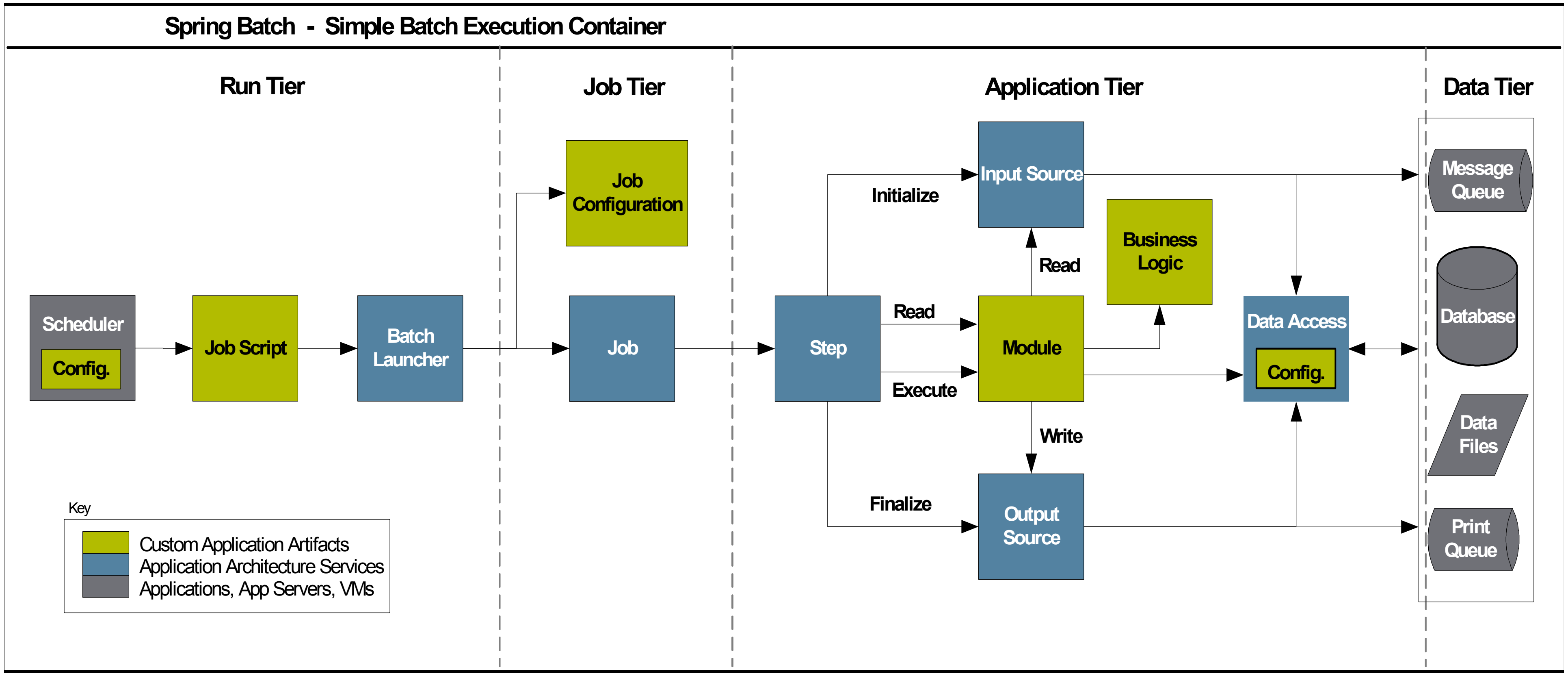
Spring Batch background

- Why is Accenture contributing to open source?
  - Collecting many years of experience with batch
  - Want to standardize batch implementation
- Why Spring?
  - Established and active community with regular releases.
  - A part of many/most Java projects
- Goal
  - A standarized, scalable batch architecture that is easy to use.

Spring Batch: Scenarios

- Periodical commitment of chunks
- Manual or planned restart after errors
- Sequential processing of dependent steps
- Parallel processing

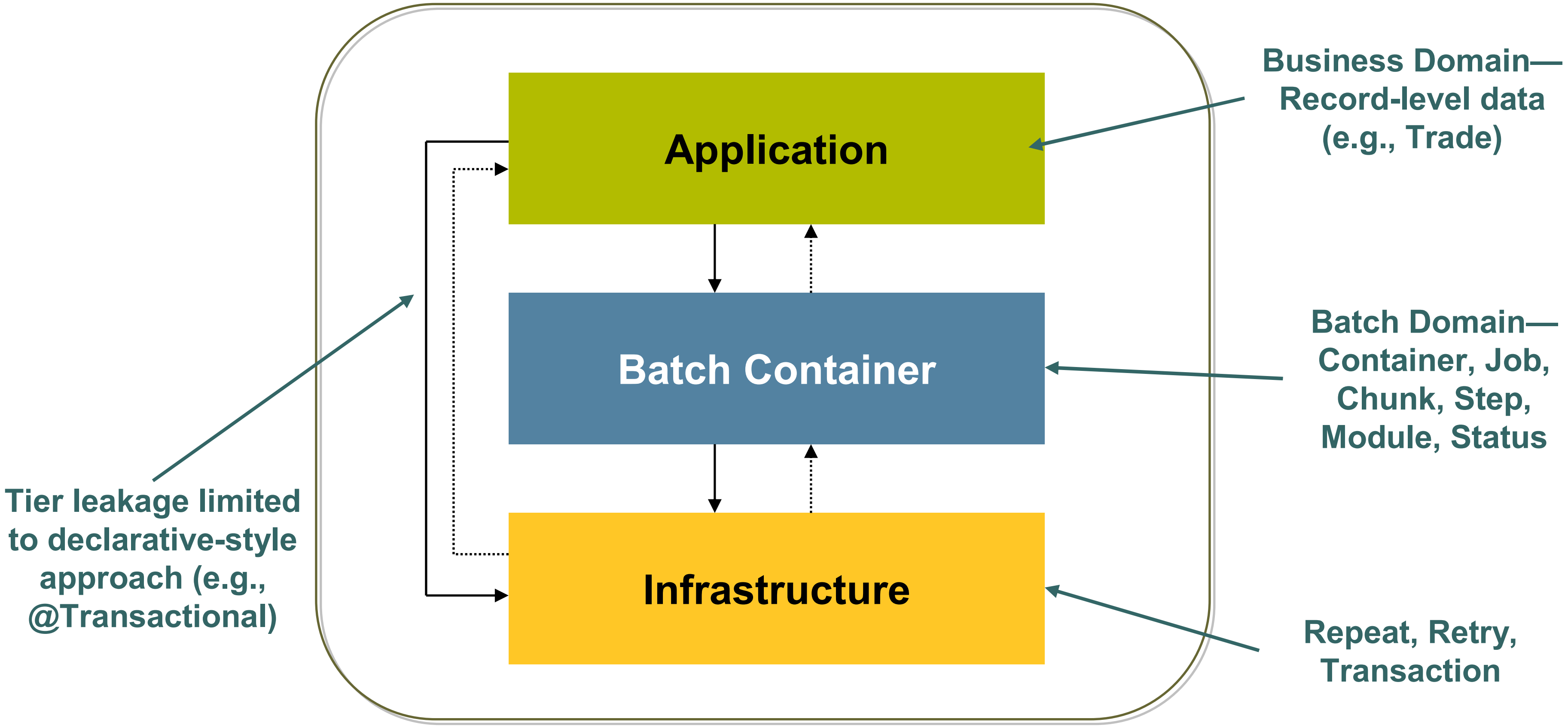
Batch reference model



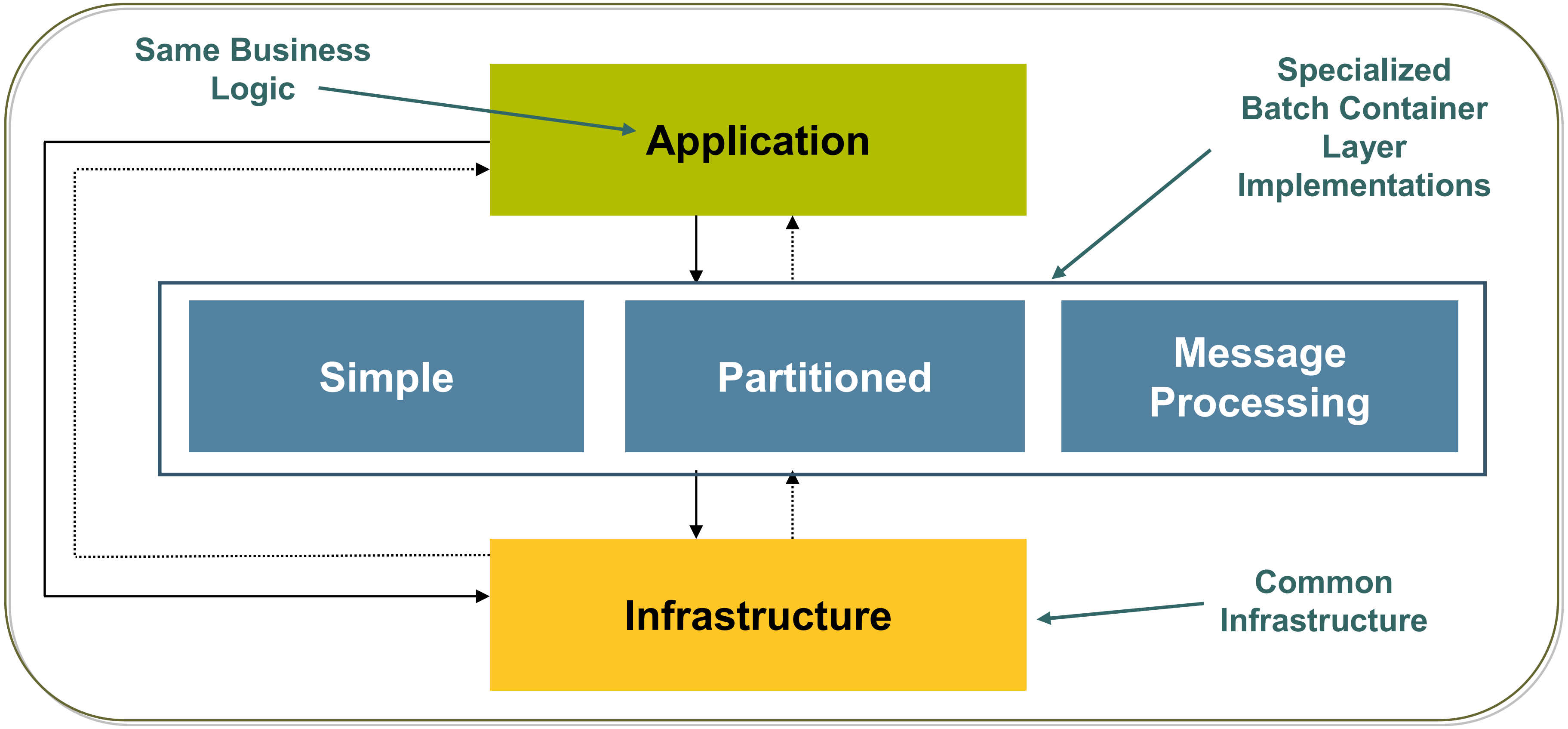
Spring Batch: Features

- Built in support of different formats
  - fixed length, delimited, XML…
- Automatic retry (policy driven)
- Execution status and statistics while batch is running and historical
- Different ways to start batch (JMX, command line)
- Support functions like
  - logging, resource management, restart, skip, etc.

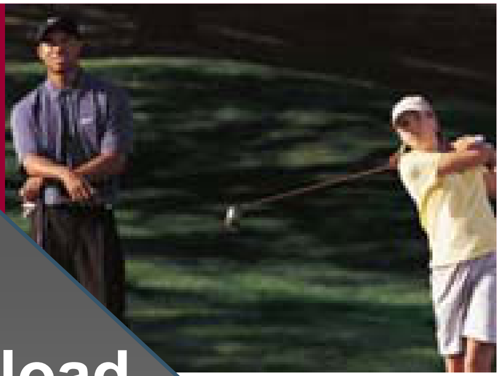
Spring Batch:
Layered architecture

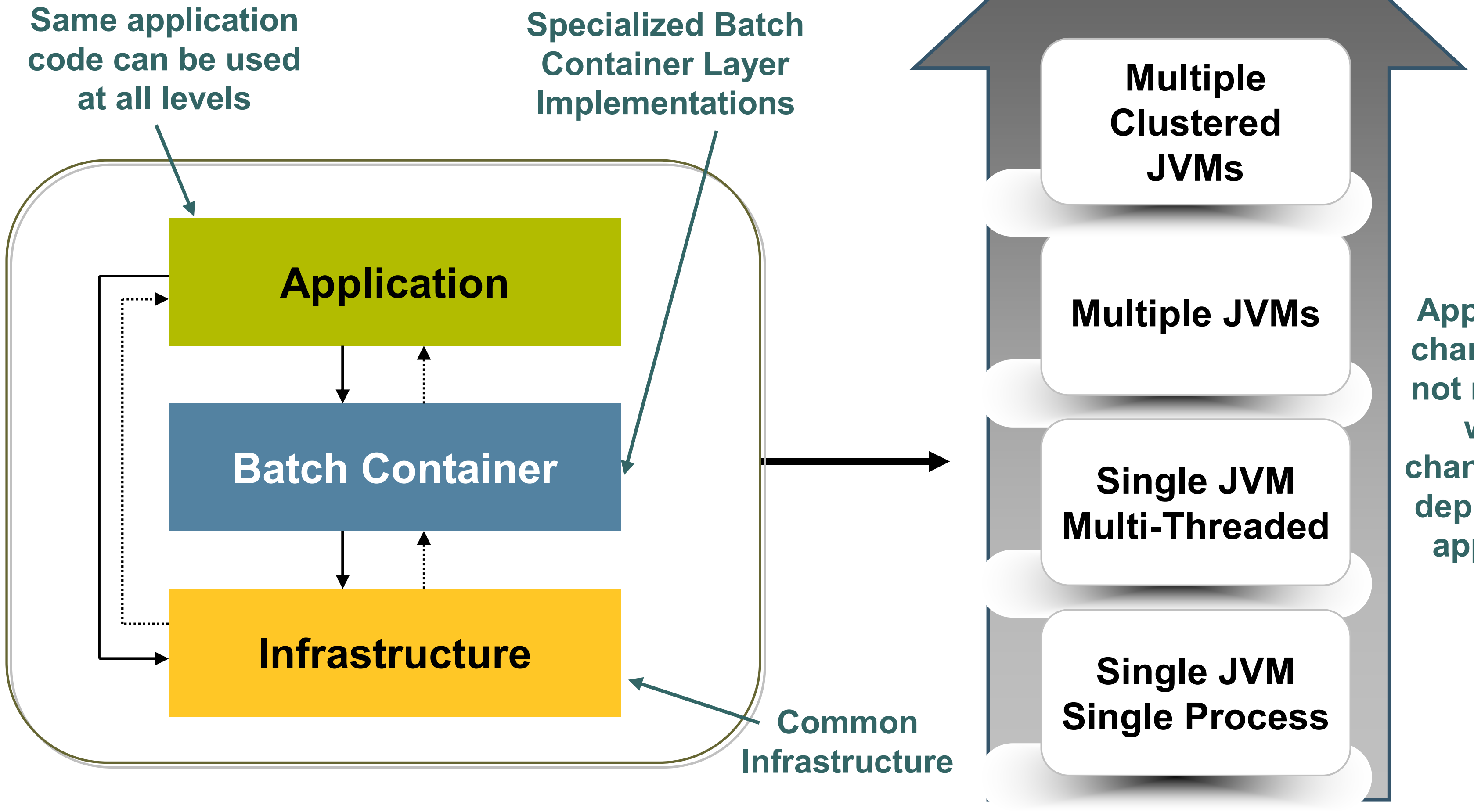


Spring Batch: different containere



Spring Batch: Scalability




Same application code can be used at all levels
Specialized Batch Container Layer Implementations
Application
Batch Container
Infrastructure
Common Infrastructure
Workload
Multiple Clustered JVMs
Multiple JVMs
Single JVM Multi-Threaded
Single JVM Single Process

Application changes are not required when changing the deployment approach

Simple Batch Pseudo Code

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

RepeatTemplate

RepeatTemplate

TransactionTemplate

Business Logic

Input and Output Retries

- Input Retry—Skip
  - Error in input data
  - Should not stop futher processing.
  - Skip row, log it and continue
- Output Retry—Recover
  - Transient errors (backend system down)
  - Will force roll back and retry.

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)

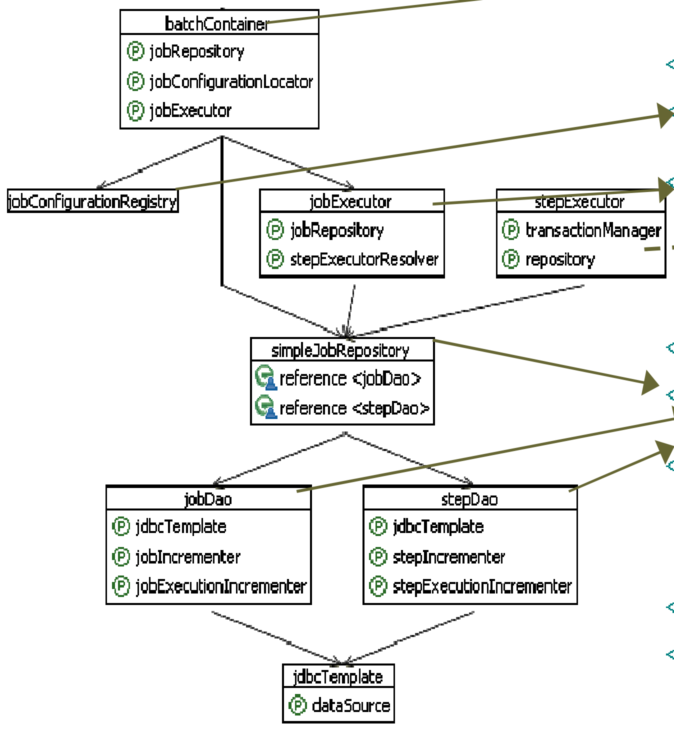
Complete normally (6)

Commit (7)

Programming the Spring way

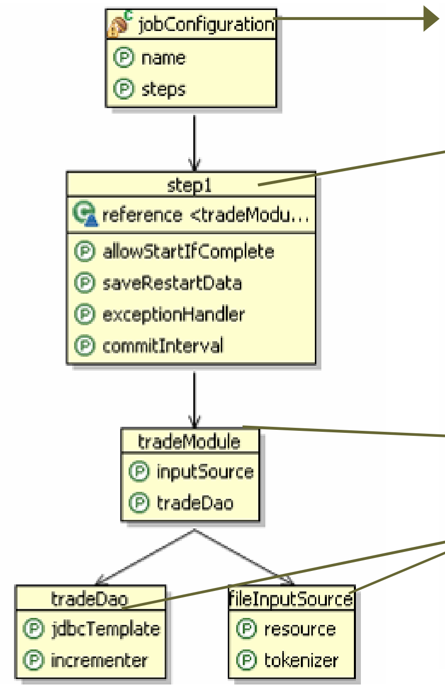
- Write business logic in POJOs
  - **`ItemProvider`**: Returns single item
  - **`ItemProcessor`**: More complex, but allows logic spanning several items.
- Postpone architecture choices
  - Implementation of batch logic is unchanged even if deployment is changes, simplifies scalability

Configuration of simple container



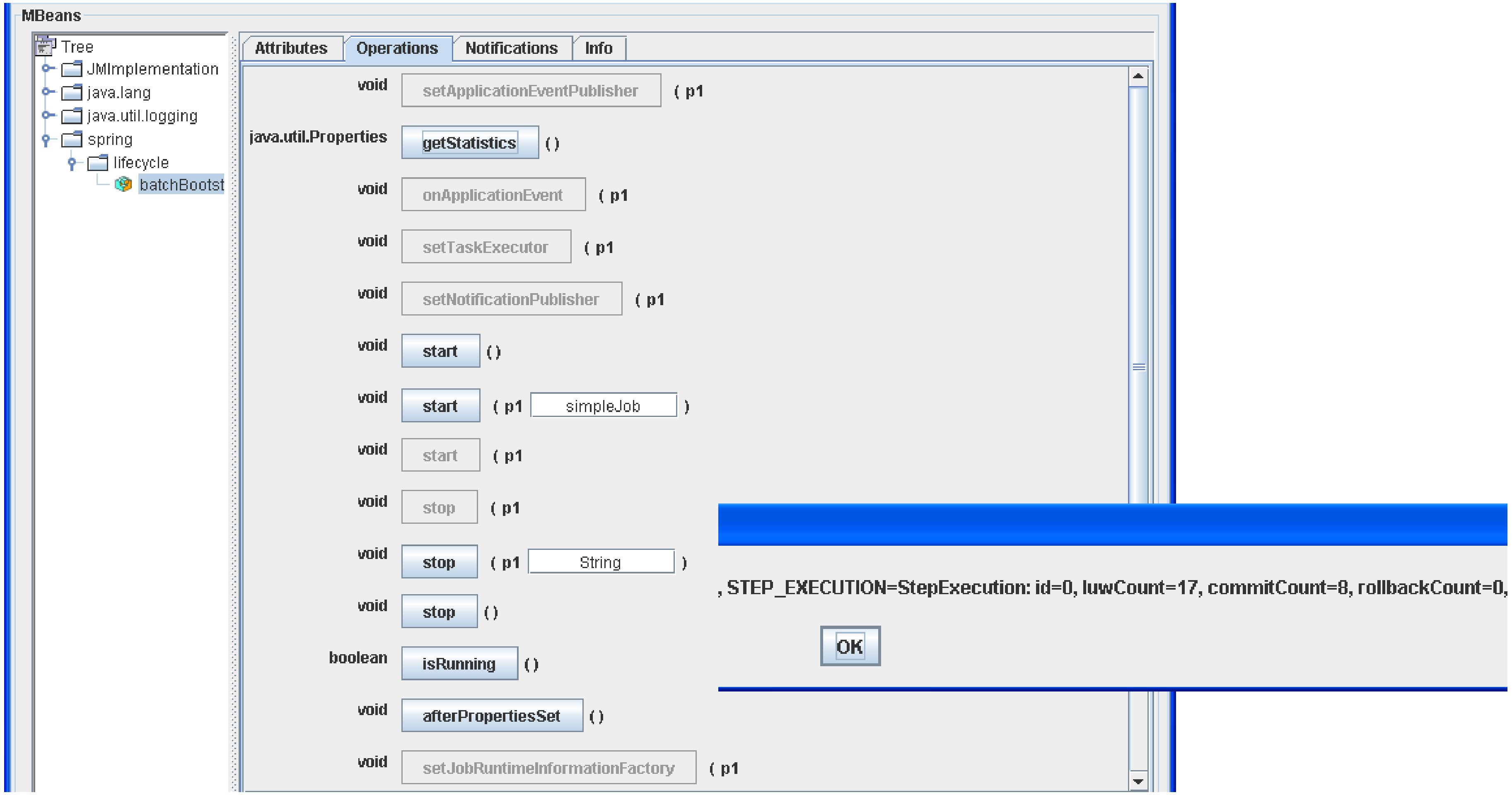
```xml
<bean id="batchContainer" class="org.springframework.batch.container.simple.SimpleBatchContainer">
  <property name="jobRepository" ref="simpleJobRepository" />
  <property name="jobConfigurationLocator" ref="jobConfigurationRegistry"/>
  <property name="jobExecutor" ref="jobExecutor" />
</bean>

<bean id="jobConfigurationRegistry"
  class="org.springframework.batch.container.common.configuration.support.MapJobConfigurationRegistry"/>

<bean id="jobExecutor" class="org.springframework.batch.container.common.executor.support.DefaultJobExecutor">
  <property name="jobRepository" ref="simpleJobRepository" />
  <property name="stepExecutorResolver">
    <bean class="org.springframework.batch.container.common.executor.support.DefaultStepExecutorResolver">
      <property name="stepExecutorName" value="stepExecutor" />
    </bean>
  </property>
</bean>

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

  <bean id="jobDao" class="org.springframework.batch.container.common.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.container.common.repository.dao.SqlStepDao">
  <property name="jdbcTemplate" ref="jdbcTemplate" />
  <property name="stepIncrementer" ref="stepIncrementer" />
  <property name="stepExecutionIncrementer" ref="stepExecutionIncrementer" />
</bean>

<bean id="stepExecutor" class="org.springframework.batch.container.common.executor.support.SimpleStepExecutor"
  scope="prototype">
  <property name="transactionManager" ref="transactionManager" />
  <property name="repository" ref="simpleJobRepository" />
</bean>
```

Configuration of simple batch job



```
<bean id="jobConfiguration" parent="simpleJob">
<property name="name" value="simpleModuleJob" />
  <property name="steps">
    <list>
      <bean id="step1" class="org.springframework.batch.container.common.configuration.StepConfigurati
        <property name="allowStartIfComplete" value="true" />
        <property name="saveRestartData" value="false" />
        <property name="exceptionHandler">
          <bean
            class="org.springframework.batch.container.common.exception.handler.SimpleLimitExceptionHa
            <property name="limit" value="5" />
            <property name="useParent" value="true"/>
          </bean>
        </property>
        <constructor-arg>
            <bean id="tradeModule" class="org.springframework.batch.sample.module.SimpleTradeModule">
              <property name="inputSource" ref="fileInputSource" />
              <property name="tradeDao" ref="tradeDao" />
            </bean>
        </constructor-arg>
        <property name="commitInterval" value="2" />
      </bean>
    </list>
```

JMX



MBeans

Tree
- JMImplementation
- java.lang
- java.util.logging
- spring
  - lifecycle
    - batchBootst

Attributes | Operations | Notifications | Info

```
void                  setApplicationEventPublisher  ( p1
java.util.Properties  getStatistics  ()
void                  onApplicationEvent  ( p1
void                  setTaskExecutor  ( p1
void                  setNotificationPublisher  ( p1
void                  start  ()
void                  start  ( p1 simpleJob )
void                  start  ( p1
void                  stop  ( p1
void                  stop  ( p1 String )
void                  stop  ()
boolean               isRunning  ()
void                  afterPropertiesSet  ()
void                  setJobRuntimeInformationFactory  ( p1
```

```
, STEP_EXECUTION=StepExecution: id=0, luwCount=17, commitCount=8, rollbackCount=0,
```

OK