SPRING BATCH OVERVIEW

Outline

- Introduction to Batch Processing
- Introduction to Spring Batch
- Architecture
- Spring Batch Concepts and Domain Model
- Create, Configure, Run and Manage Batch Jobs
- Scaling and Parallel Processing
- Batch Patterns
- Spring Batch Integration

Need for Batch Processing

- Automated, complex processing of large volumes of information without user interaction (Time Based Events)
  - Month end calculations
  - Notices, Correspondences
- Periodic application of complex business rules processed repetitively across very large data set
  - Insurance Benefit Determination
  - Rate Adjustments
- Integration of information received from internal and external systems that typically requires below operations in transactional manner
  - Formatting
  - Validation
  - Processing

Batch Processing Use Cases

- Close of business processing
  - Order processing
  - Business reporting
  - Account reconciliation
- Import/export handling
  - Instrument/position import
  - Trade/allocation export
- Large-scale output jobs
  - Loyalty scheme emails
  - Financial statements

Batch Processing 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 Overview

- Lightweight, comprehensive batch framework to develop robust batch apps
- Build on top of Spring Framework – Leverages IoC container and POJO based development
- Spring Batch is NOT a scheduling framework – can work with Quartz, Tivoli, Control-M
- Provides reusable functions essential for processing large volumes of records
  - Logging / Tracing
  - Transaction management
  - Job processing statistics
  - Job restart, Skip
  - Resource management
- Support for scaling, parallel processing and partitioning to enable large scale batch processing

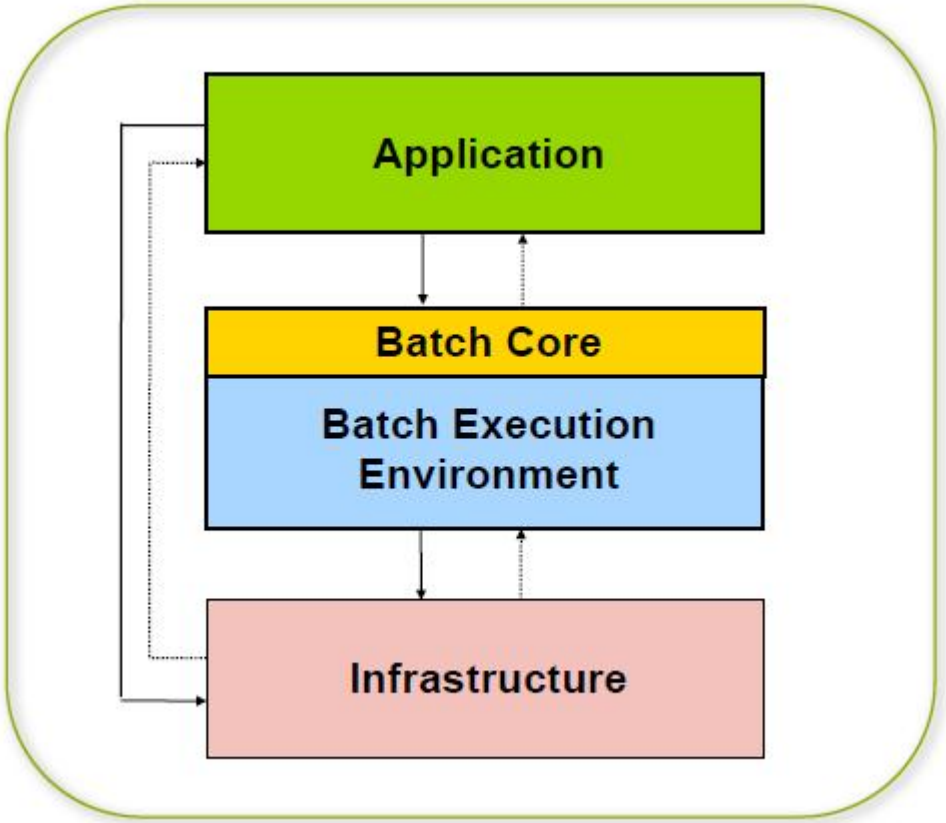
Spring Batch Technical Objectives

- Batch developers use the Spring programming model: Concentrate on business logic and let the framework take care of infrastructure.
- Clear separation of concerns between the infrastructure, the batch execution environment, and the batch application.
- Provide common, core execution services as interfaces that all projects can implement.
- Provide simple and default implementations of the core execution interfaces that can be used 'out of the box'.
- Easy to configure, customize, and extend services, by leveraging the spring framework in all layers.
- All existing core services should be easy to replace or extend, without any impact to the infrastructure layer.
- Provide a simple deployment model, with the architecture JARs completely separate from the application, built using Maven.

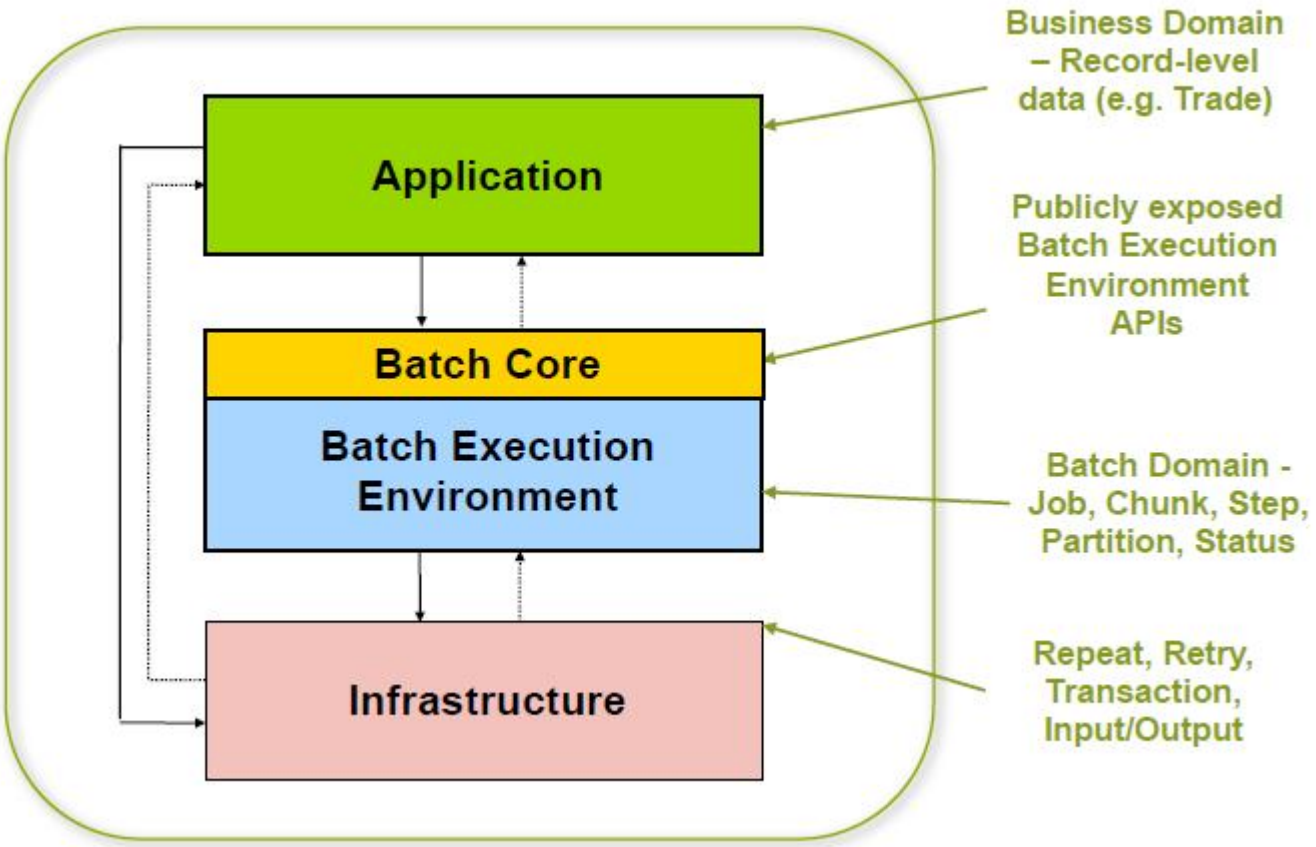
Spring Batch Features

- 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.

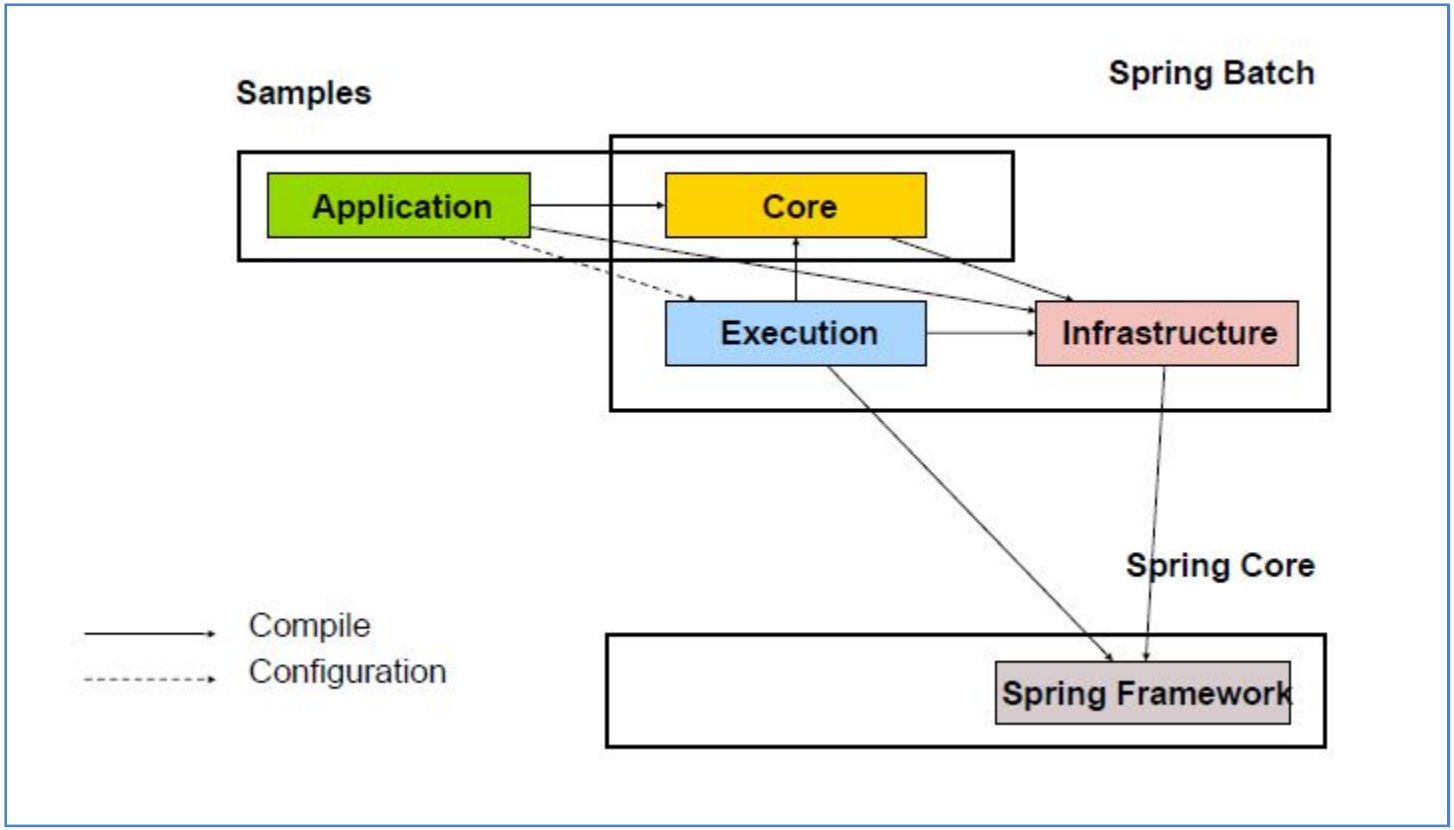
Spring Batch Architecture



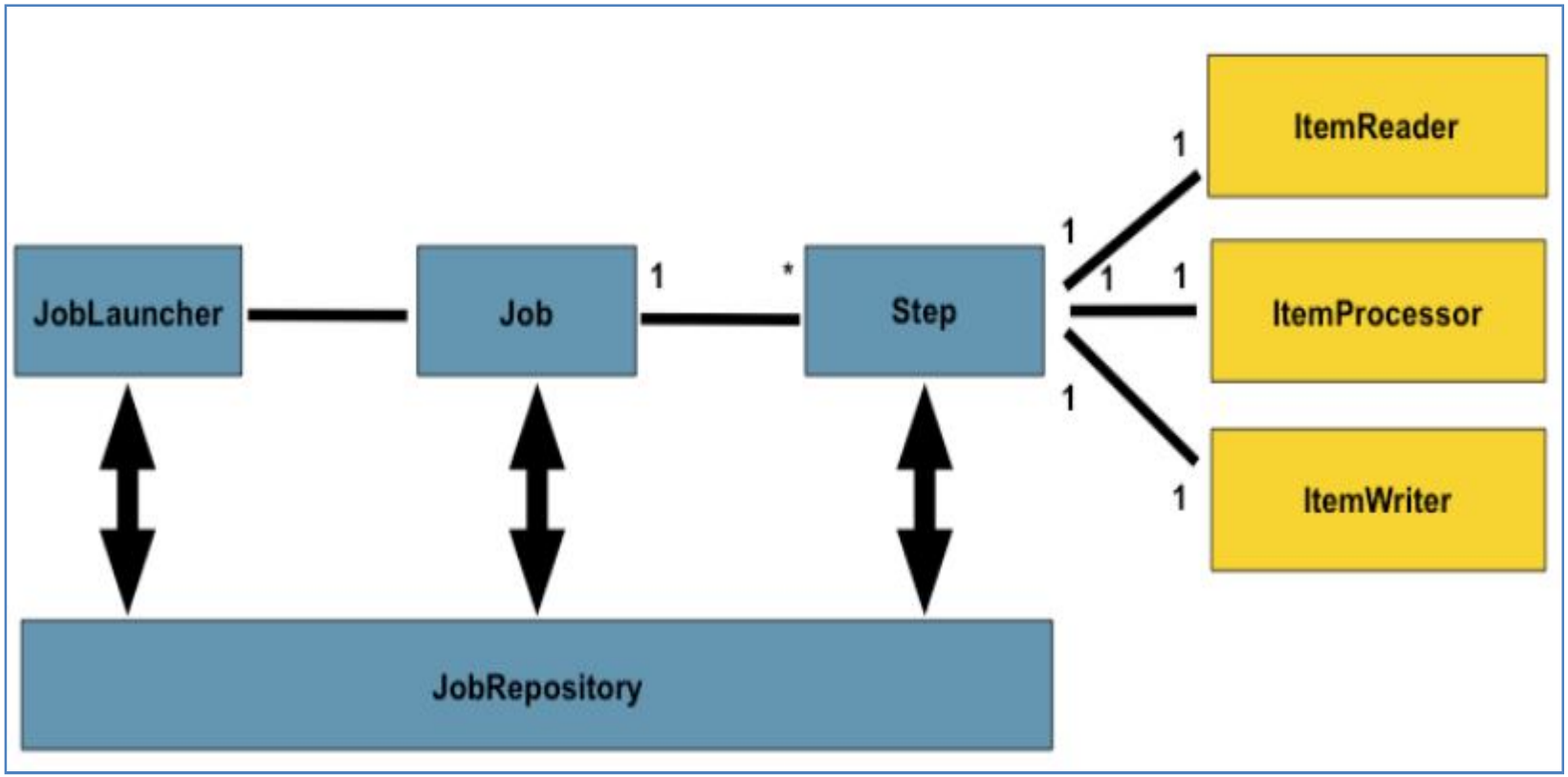
Spring Batch Architecture



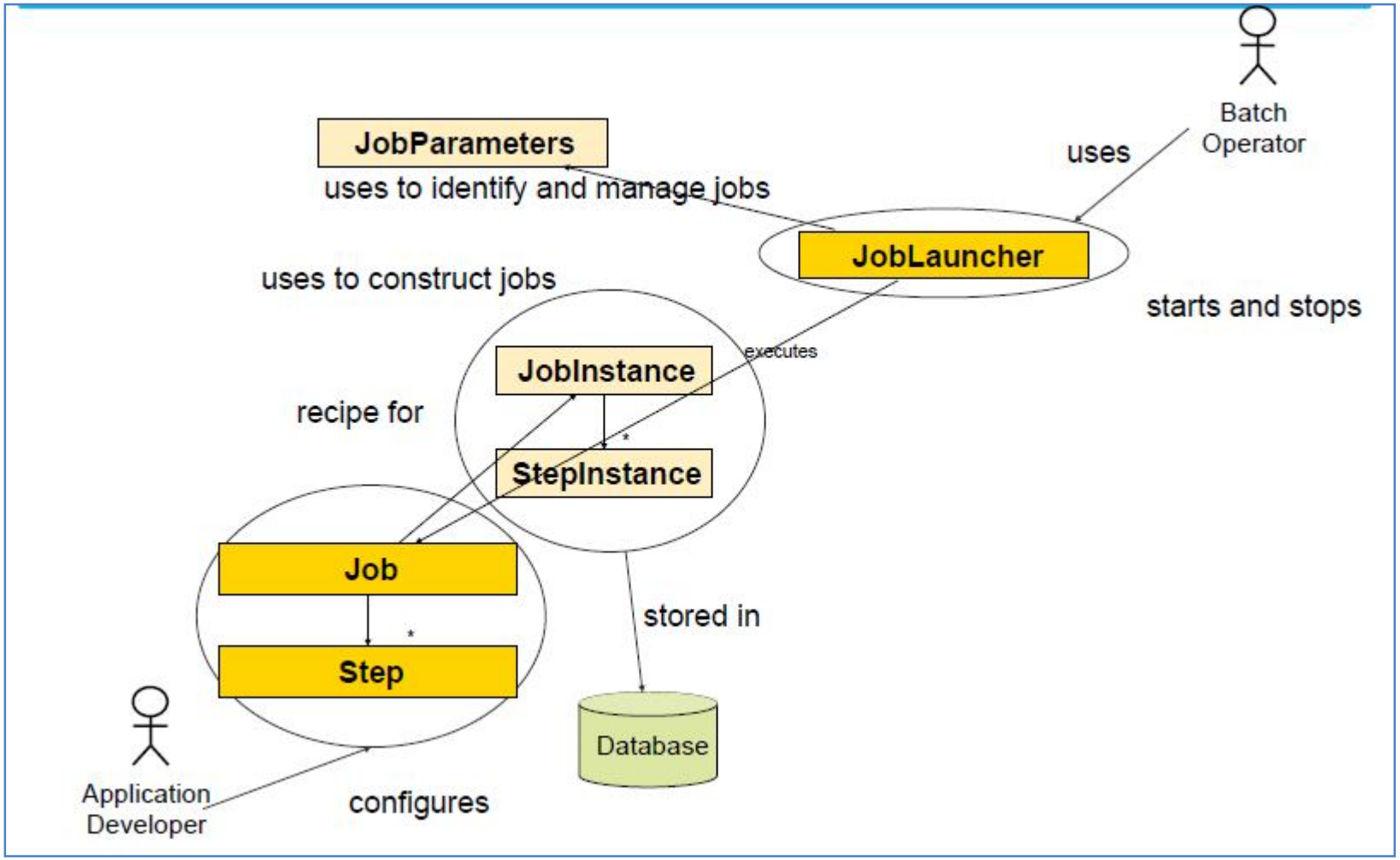
Spring Batch Depedencies



Spring Batch Domain Components



Spring Batch Domain Model

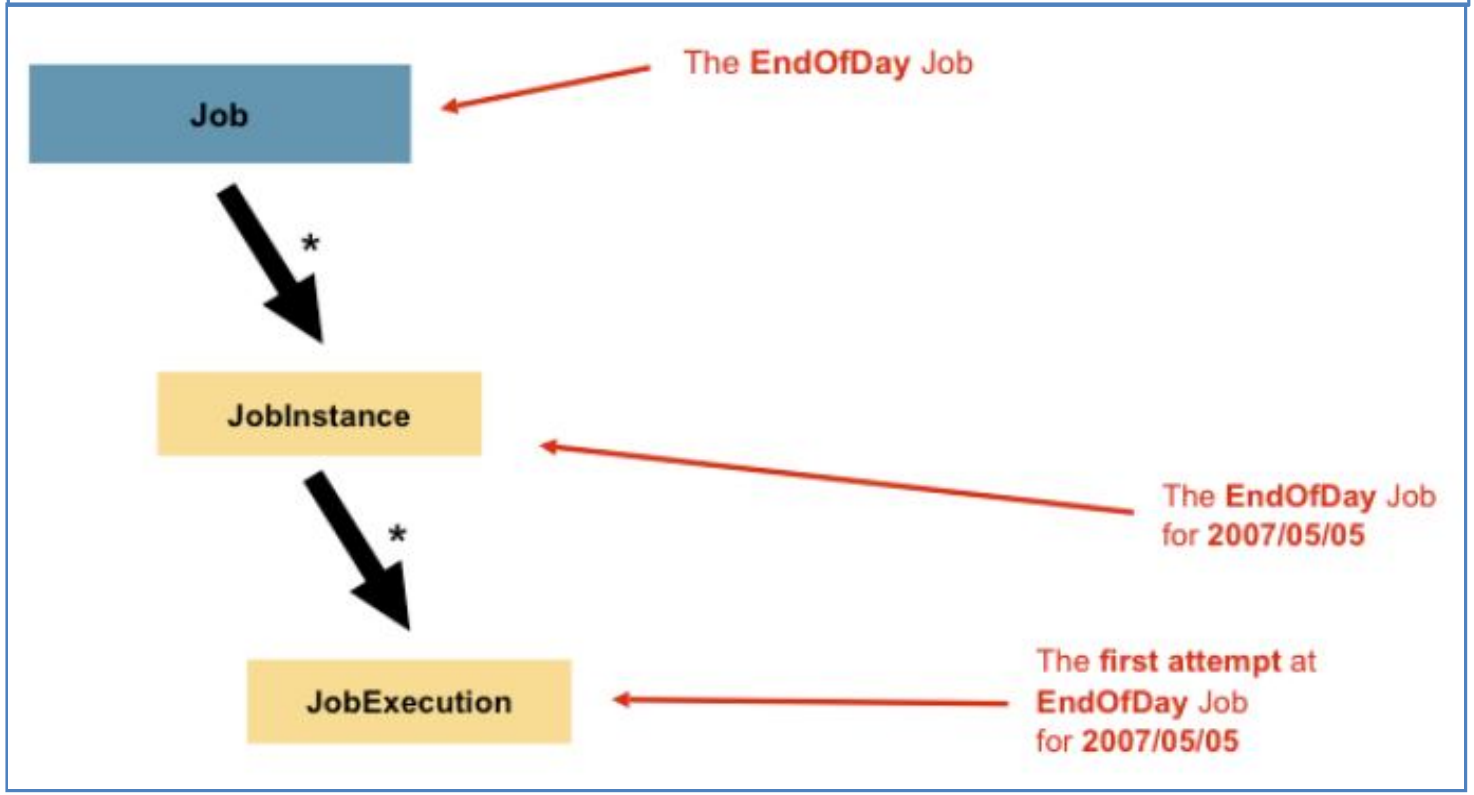


Job, Job Instance, Job Execution

Job is an entity that encapsulates an entire batch process

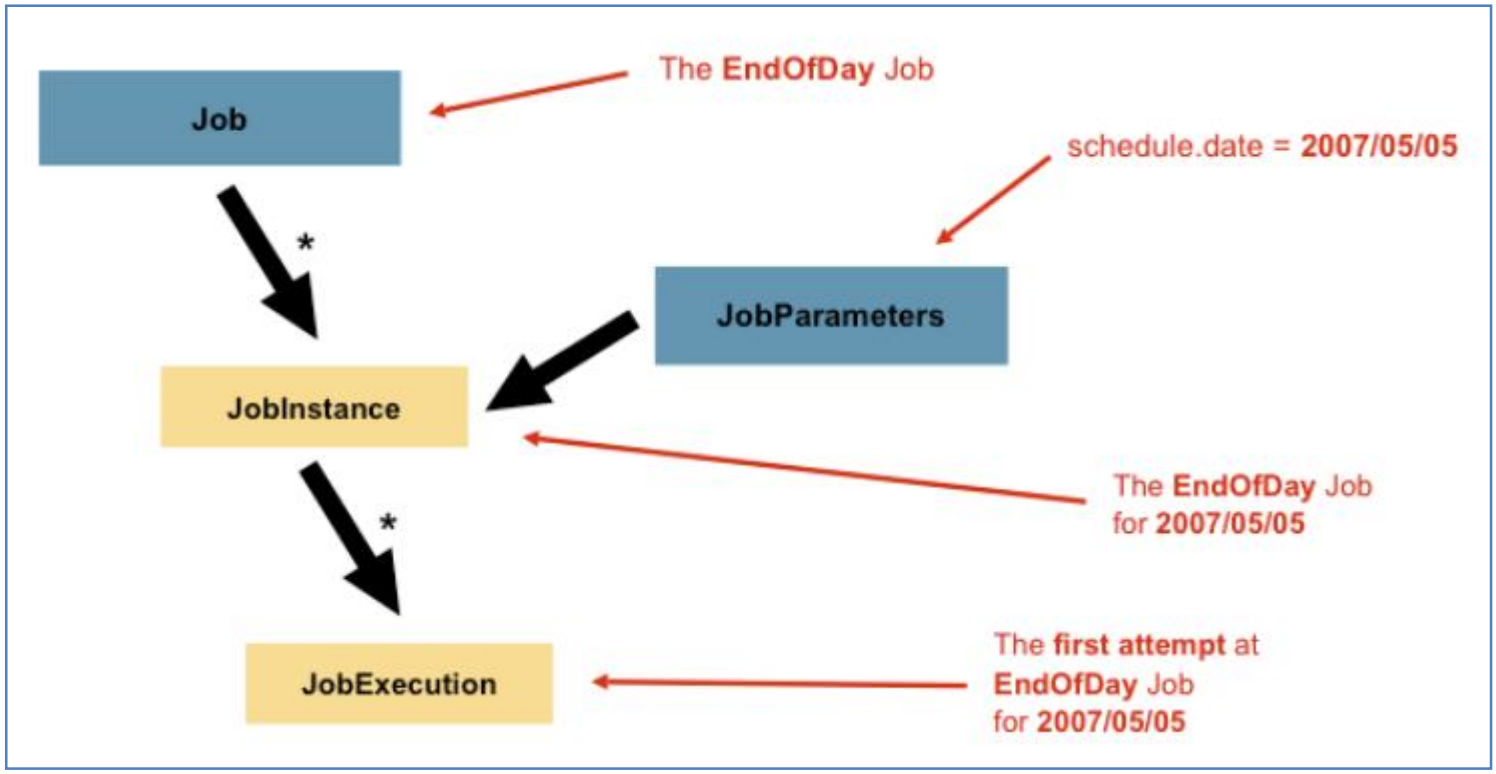
JobInstance refers to the concept of a logical job run

JobExecution refers to single attempt to run a Job



Job Parameters

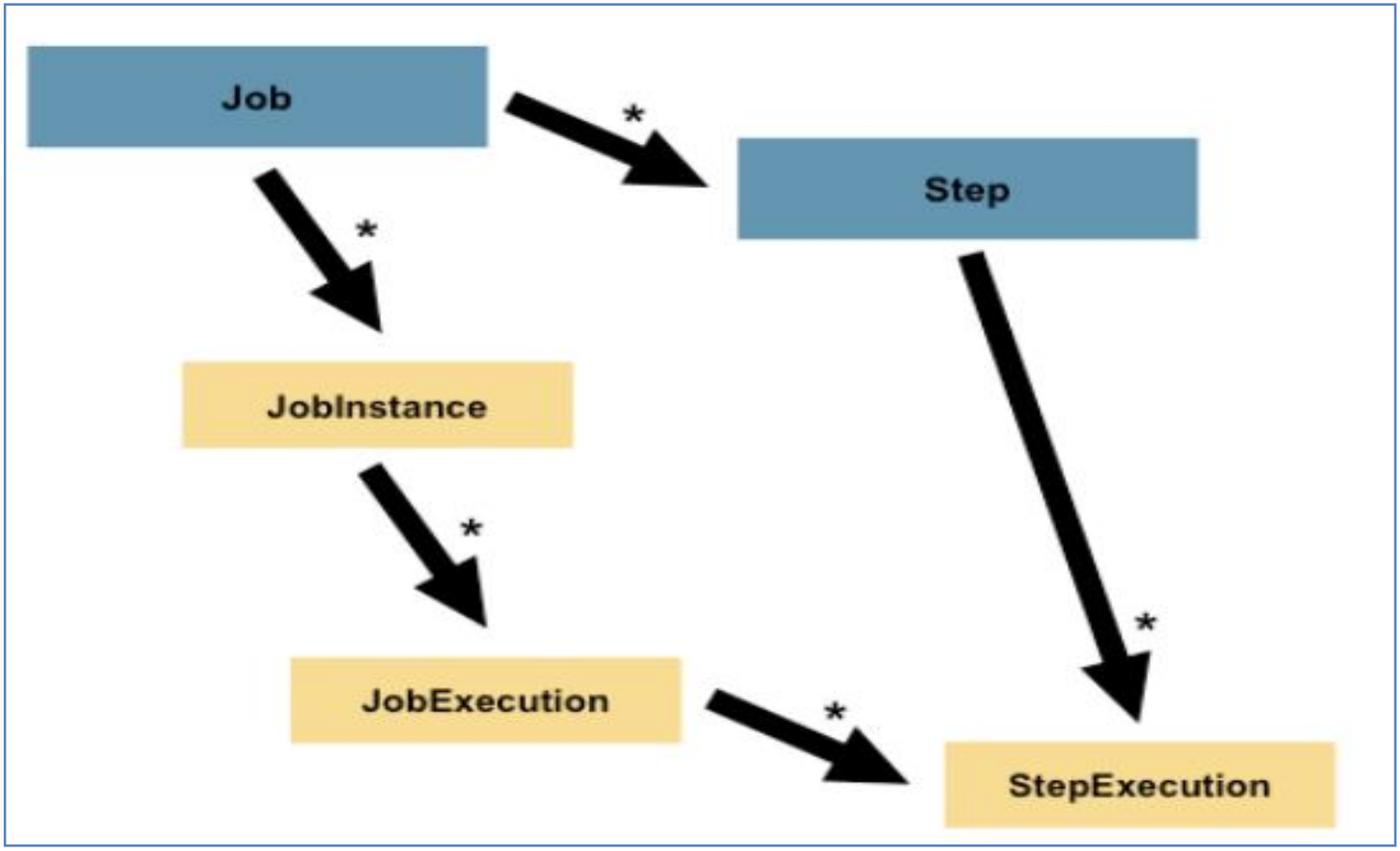
Job Parameters object holds a set of parameters used to start a batch job



Step, Step Execution

Step is a domain object that encapsulates an independent, sequential phase of a batch job

StepExecution represents a single attempt to execute a Step



ExecutionContext, JobRepository, JobLauncher

ExecutionContext represents a collection of key/value pairs that are persisted and controlled by the framework in order to allow developers a place to store persistent state that is scoped to a StepExecution object or a JobExecution object

JobRepository is the persistence mechanism for all batch domain objects and provides CRUD operations for JobLauncher, Job, Step implementations.

When a Job is first launched, a JobExecution is obtained from the repository, and, during the course of execution StepExecution and JobExecution implementations are persisted by passing them to the repository.

JobLauncher represents a simple interface for launching a Job with a given set of JobParameters

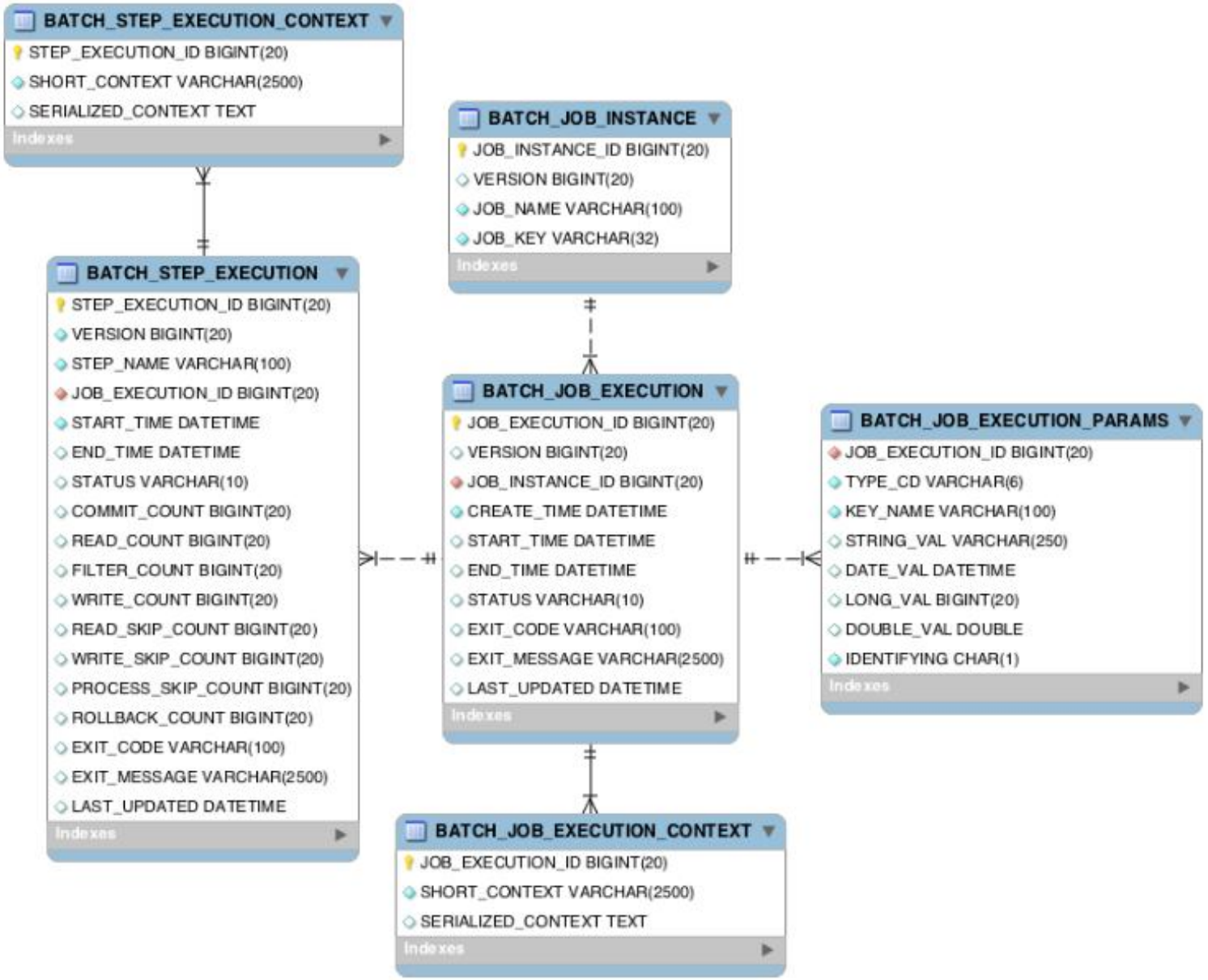
ItemReader, ItemWriter, ItemProcessor

ItemReader is an abstraction that represents the retrieval of input for a Step, one item at a time

ItemReader is an abstraction that represents the retrieval of input for a Step, one item at a time

ItemProcessor is an abstraction that represents the business processing of an item

Spring Batch Metadata Schema



Batch Design Principles and Guidelines

- Design batch architecture to reuse existing components and services
- Create multiple simple batch jobs over single complex batch job
- Keep data storage and processing physically closer
- Minimize system resource use, especially I/O
- Review application I/O to ensure unnecessary physical I/O is avoided
- Do not do things twice in batch run
- Allocate enough memory at the beginning
- Ensure adequate checks and validation for data integrity
- Plan and execute stress tests at earliest in prod grade environment
- Ensure proper backup strategies are in place

Batch Processing Strategies

- Conversion Applications
- Validation Applications
- Extract Applications
- Extract/Update Applications
- Processing and Updating Applications
- Output/Format Applications

Thank You!