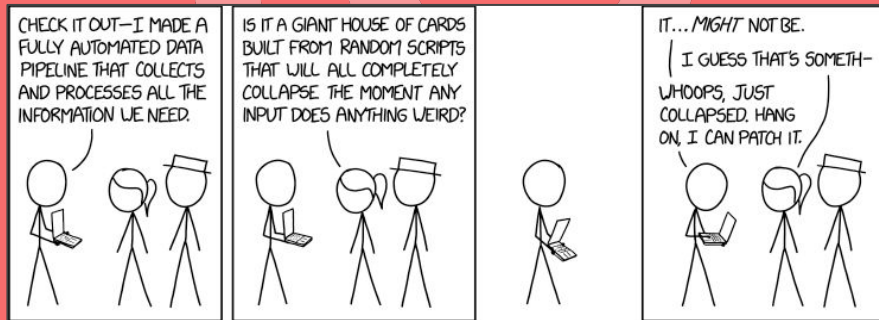


Batch Processing in Modern Java

Data bulk processing with Spring Batch



- ❖ What is Batch Processing?
- ❖ Use Cases for Batch Processing in Enterprises
- ❖ JSR 352 and Spring Batch
- ❖ Enriching People: Demo for Spring Batch
- ❖ Maintaining Batch Processes with Spring Data Flow



What is Batch Processing?

“Bulk processing to perform business operations in mission critical environments”

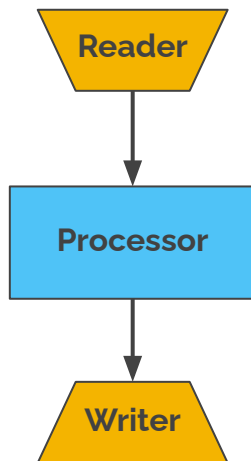
- ❖ **Automated, complex processing** of **large volumes** of information that is most efficiently processed **without user interaction**.
- ❖ **Periodic application** of complex **business rules processed repetitively** across very large data sets.
- ❖ **Integration of information** that is received from internal and external systems that typically requires formatting, validation, and processing in a **transactional manner into the system of record**.



Usage Scenarios of Batch Processing

A typical batch application generally:

- ❖ **Reads** a large number of records from a database, file, or queue.
- ❖ **Processes** the data in some fashion.
- ❖ **Writes** back data in a modified form.





What are Quality (non-functional) Requirements of Batch Processing?

“Batch processing is used to process billions of transactions every day for enterprises.”

- ❖ **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 (with extensions to workflow-driven batches)
- ❖ **Partial processing**: skip records (for example, on rollback)



Batch Processing Options

- ❖ Normal processing during a **batch window** in **off-line mode**.
- ❖ **Concurrent** batch or **on-line** processing.
- ❖ **Parallel processing** of many different batch runs or jobs **at the same time**.
- ❖ Partitioning (processing of many instances of the **same job at the same time**).
- ❖ A combination of the preceding options.



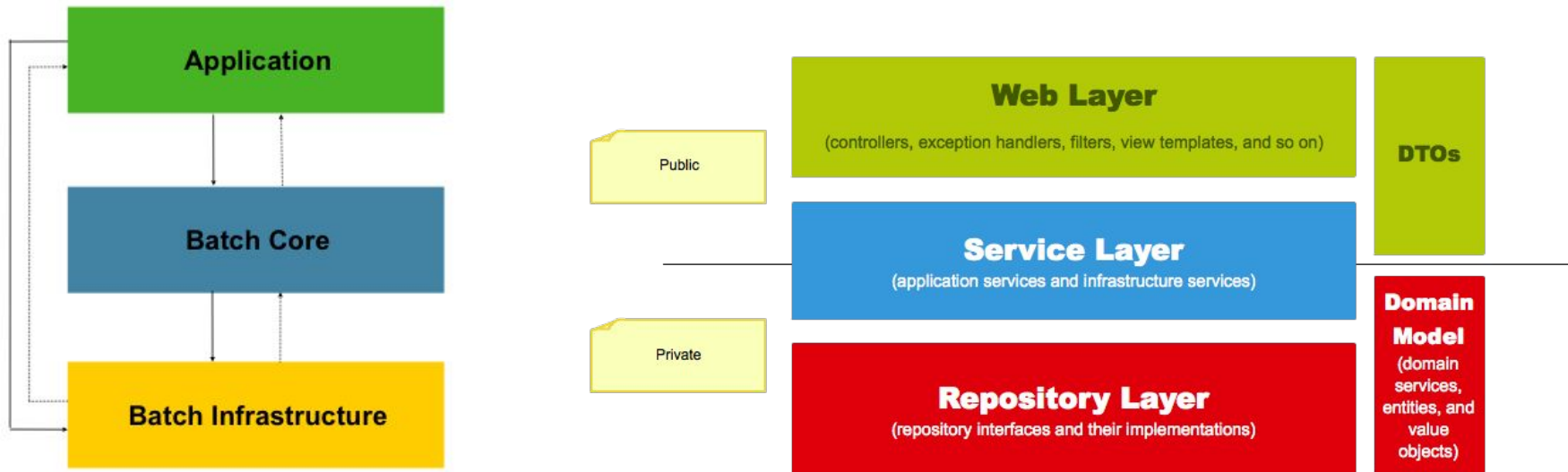
Spring Batch Functionality

- ❖ 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 or Gradle.



Spring Batch Architecture

Spring Batch Component Model is layered, similar to Web Applications, to facilitate **Seperation of Concerns** and **Reusability**.



The **Batch Core** contains the core runtime classes necessary to launch and control a batch job. It includes implementations for JobLauncher, Job, and Step. Both **Application** and **Core** are built on top of a common **infrastructure**. This **infrastructure** contains common readers and writers and services (such as the RetryTemplate), which are used both by application developers (readers and writers, such as ItemReader and ItemWriter) and the core framework itself (Retry, which is its own library).



Start with Spring Initializr



Project

☒ Maven Project ☐ Gradle Project

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 3.0.0 (SNAPSHOT) ☐ 3.0.0 (M1) ☐ 2.7.0 (SNAPSHOT) ☐ 2.7.0 (M1)
☐ 2.6.4 (SNAPSHOT) ☒ 2.6.3 ☐ 2.5.10 (SNAPSHOT) ☐ 2.5.9

Dependencies

Spring Batch I/O

Batch applications with transactions, retry/skip and chunk based processing.

ADD DEPENDENCIES... CTRL + B

[< ALL GUIDES](#)

Creating a Batch Service

This guide walks you through the process of creating a basic batch-driven solution.

What You Will build

You will build a service that imports data from a CSV spreadsheet, transforms it with custom code, and stores the final results in a database.

[Get the Code](#)[Go To Repo](#)

Projects

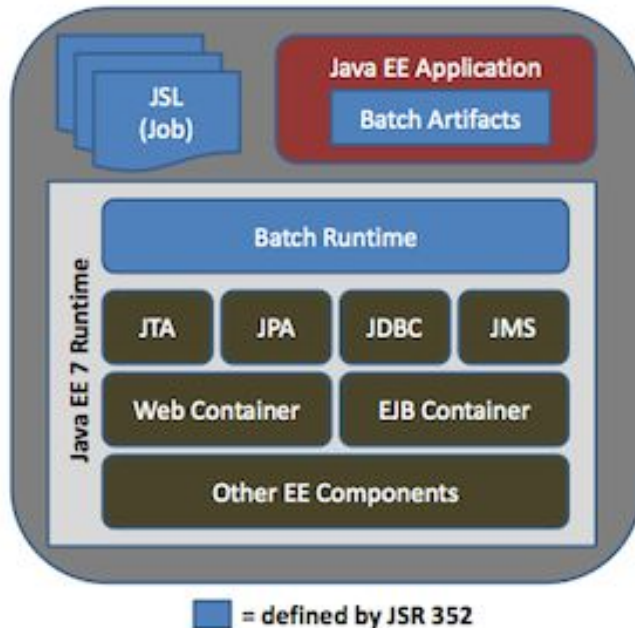
[Spring Batch](#)



What is JSR 352?

JSR 352 defines a **Job Specification Language (JSL)** to define batch jobs, **a set of interfaces** that describes the artifacts that comprise the batch programming model to implement batch business logic, and a **batch runtime** for running batch jobs, according to a defined life cycle.

- As of Spring Batch 3.0 support for JSR-352 has been fully implemented
- Spring Batch and JSR-352 are structurally the same
- Final Standard JSR 352 was included in JEE 7



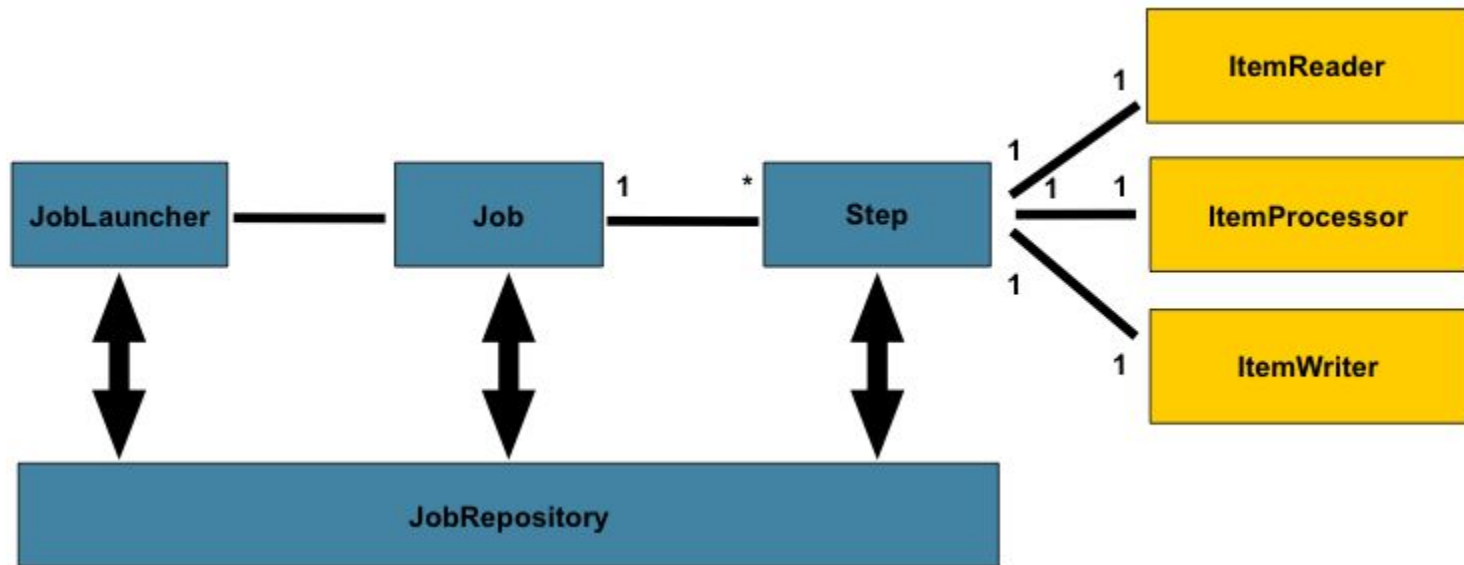


Domain Language of Batch

- ❖ Significant improvement in adherence to a **clear separation of concerns**.
- ❖ Clearly delineated architectural layers and **services provided as interfaces**.
- ❖ Simple and default implementations that allow for quick adoption and ease of use out-of-the-box.
- ❖ Significantly enhanced extensibility.



Domain Language for Batch Processing



A **Job** has one to many **steps**, each of which has exactly one **ItemReader**, one **ItemProcessor**, and one **ItemWriter**. A job needs to be launched (with **JobLauncher**), and metadata about the currently running process needs to be stored (in **JobRepository**).



“Jobs” and “Steps”

In Spring Batch, a Job is simply a container for Step instances.

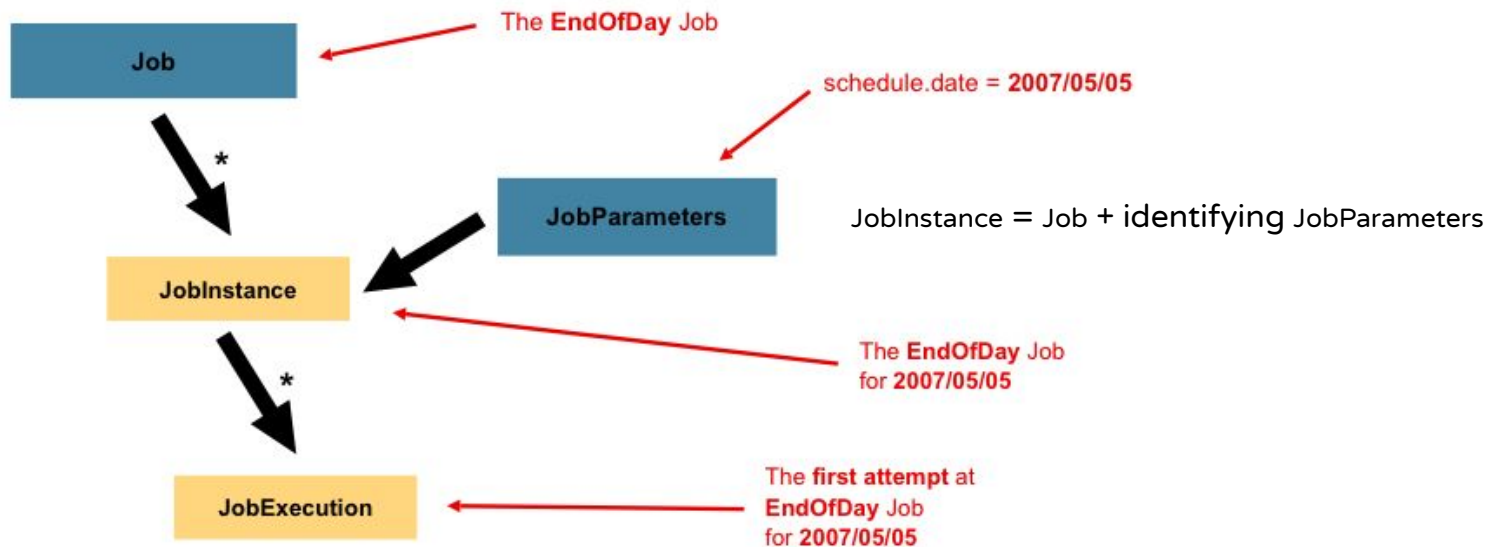
It combines **multiple steps** that belong logically together in a flow and allows for configuration of properties global to all steps, such as restartability. The job configuration contains:

- The simple name of the job.
- Definition and ordering of Step instances.
- Whether or not the job is restartable.

```
@Bean
public Job footballJob() {
    return this.jobBuilderFactory.get("footballJob")
        .start(playerLoad())
        .next(gameLoad())
        .next(playerSummarization())
        .build();
}
```



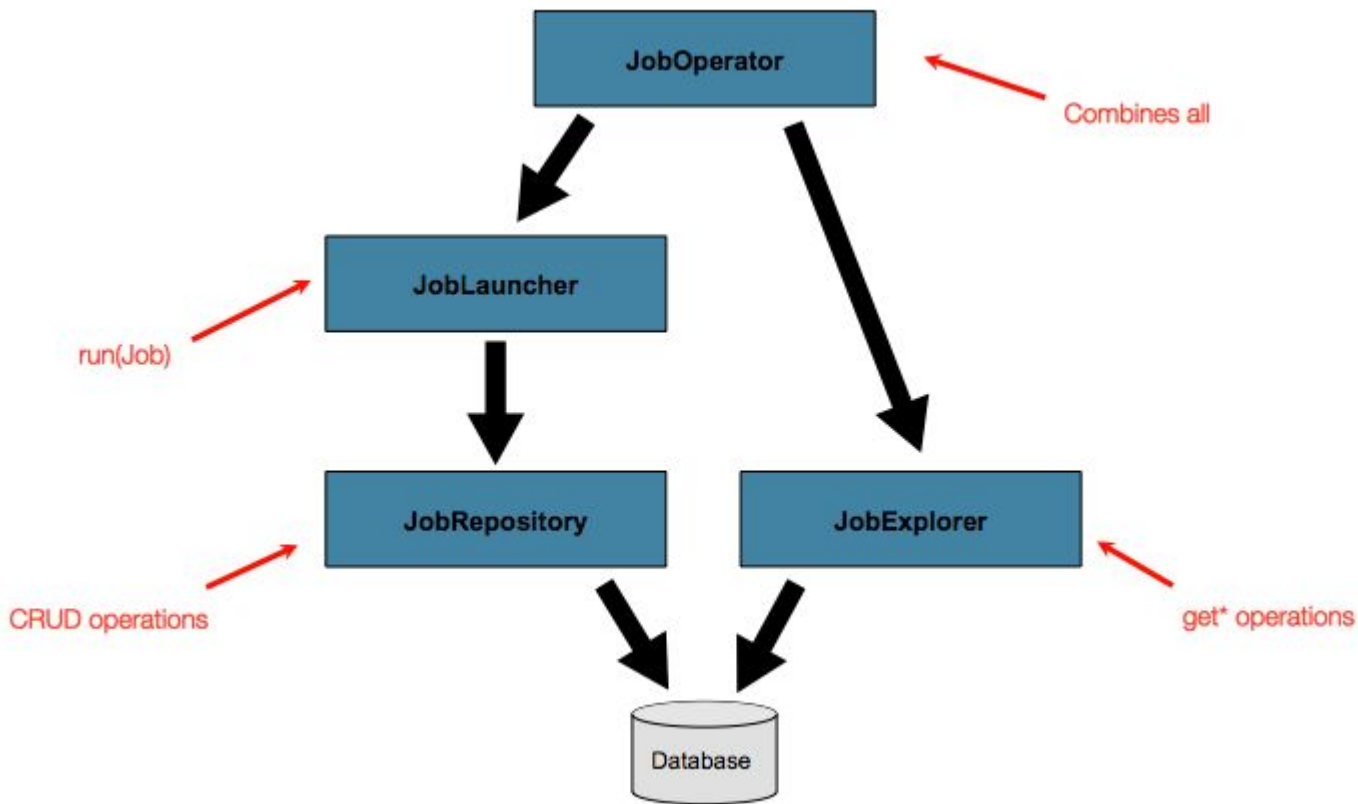
“Jobs”, “JobInstances” and “JobExecutions”



A JobExecution refers to the technical concept of a single attempt to run a Job

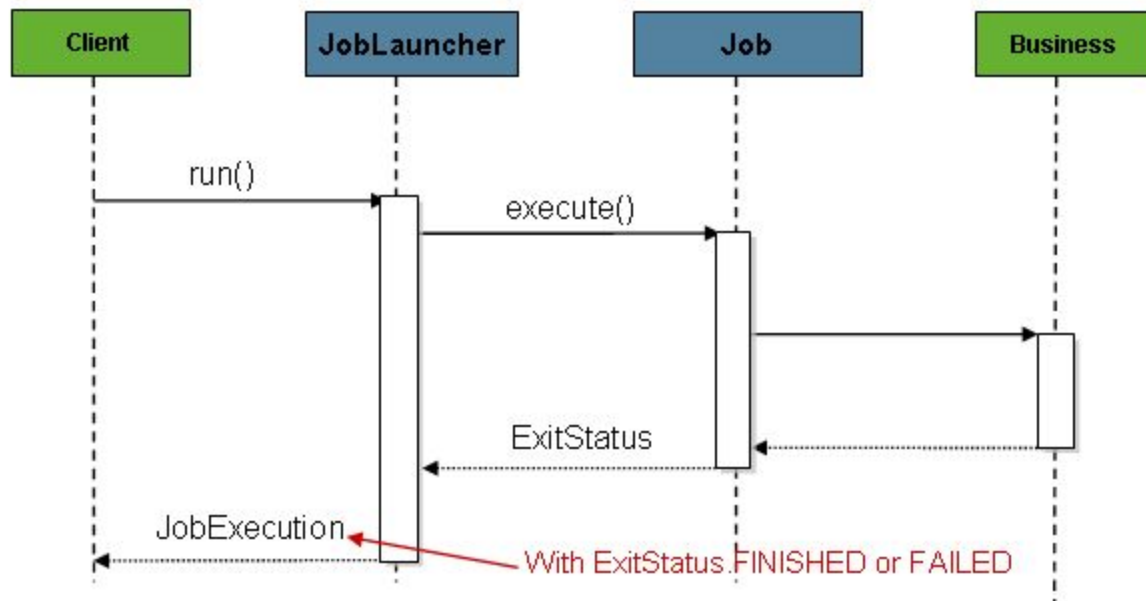


Batch Job Maintenance and Persistence



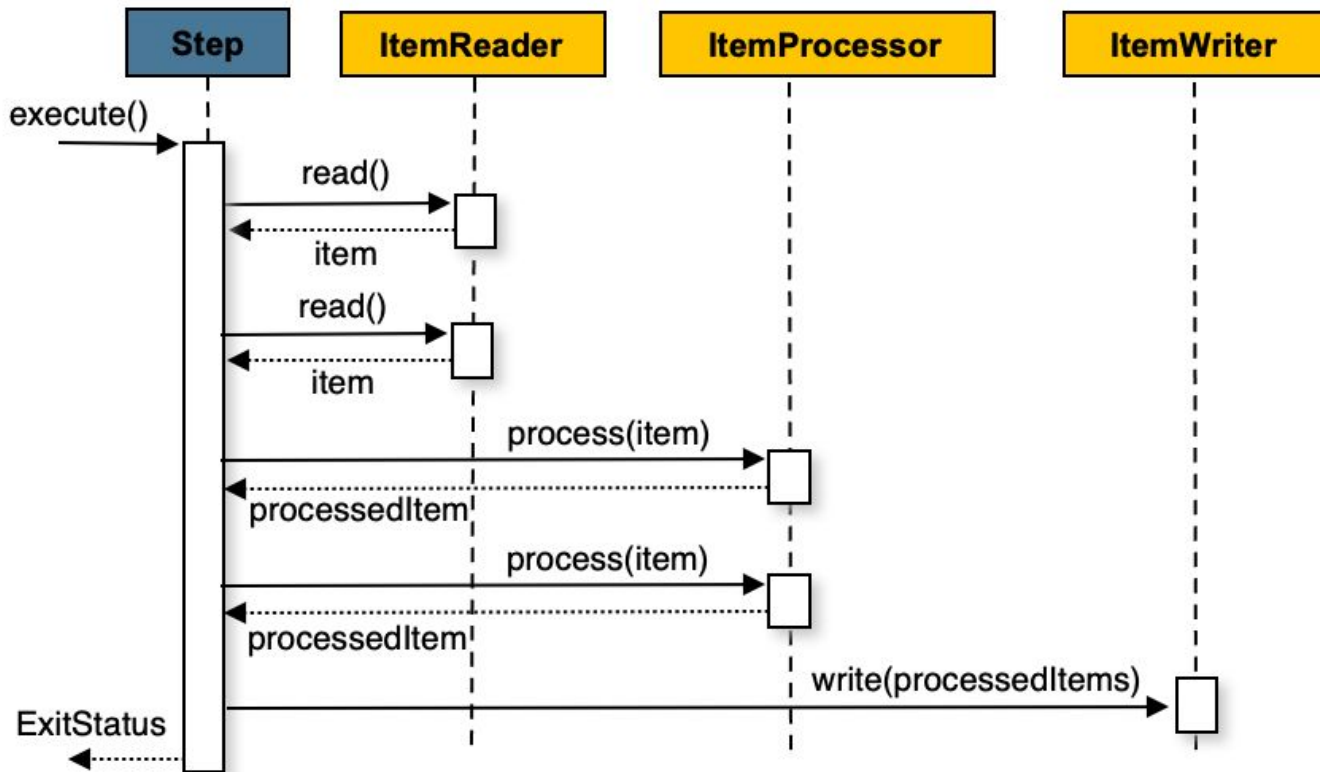


Batch Processing Flow from Client



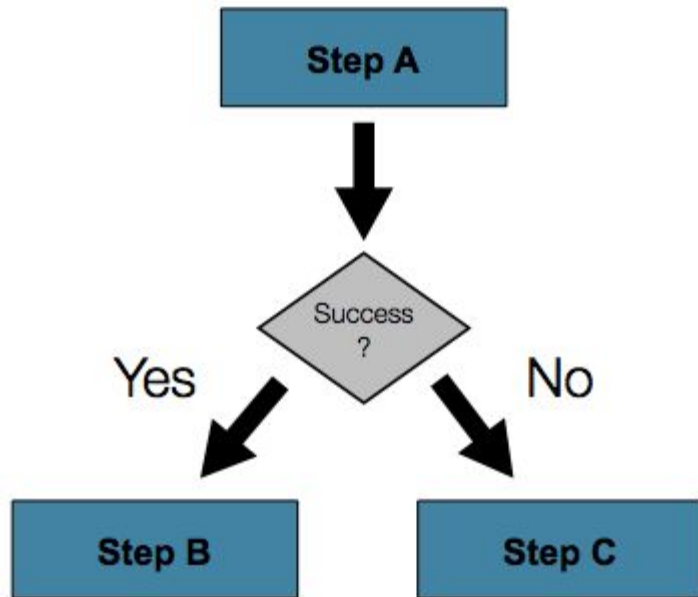
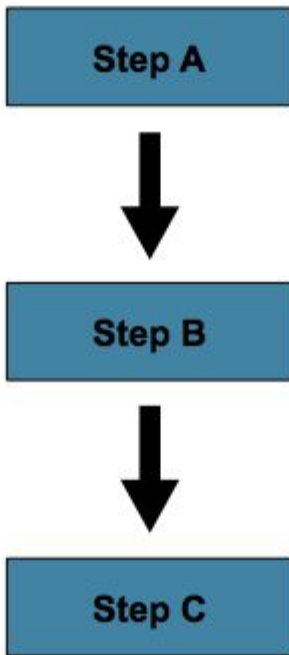


Batch Processing Internal Flow



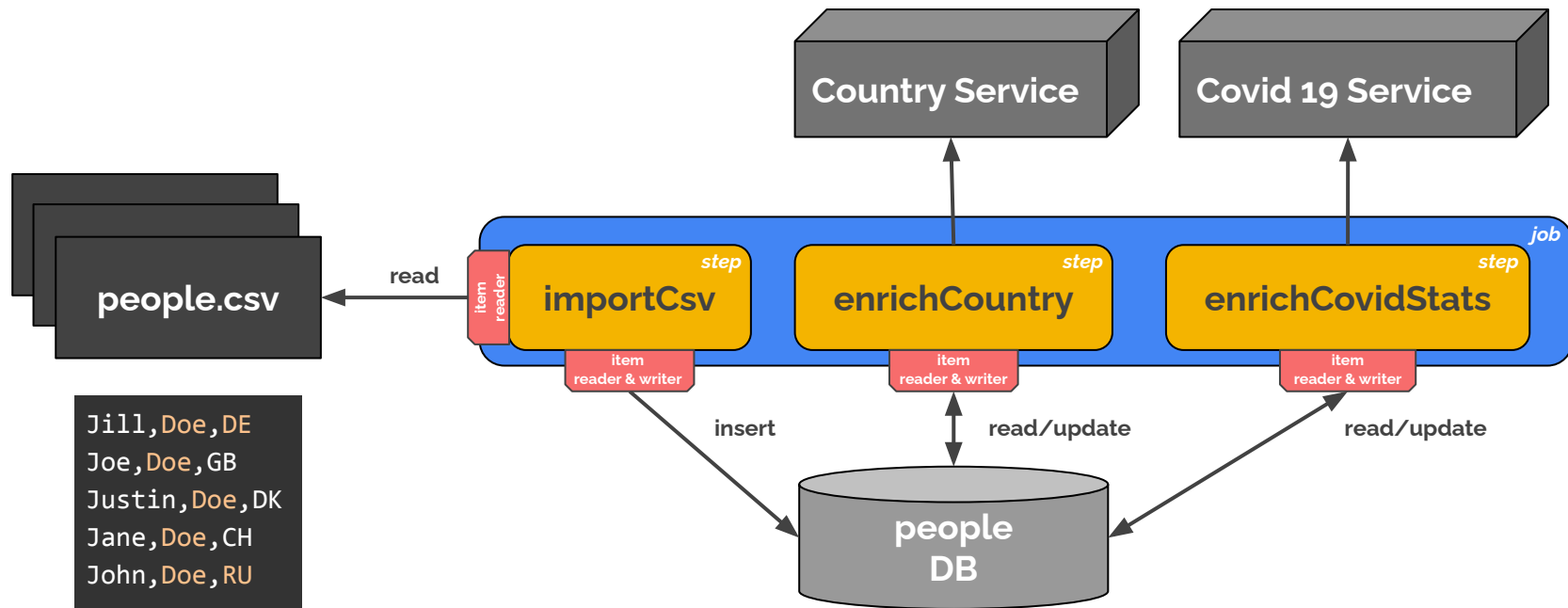


Sequential vs Conditional Flows





Demo: People Enrichment Batch Job



Jill,Doe,DE
Joe,Doe,GB
Justin,Doe,DK
Jane,Doe,CH
John,Doe,RU

FIRST_NAME	LAST_NAME	COUNTRY	CAPITAL	POPULATION	ACTIVE_COVID_CASES
JILL	DOE	DE	["Berlin"]	83240525	10025463
JOE	DOE	GB	["London"]	67215293	163
JUSTIN	DOE	DK	["Copenhagen"]	5831404	1742569
JANE	DOE	CH	["Bern"]	8654622	2220539
JOHN	DOE	RU	["Moscow"]	144104080	11670366