# Migration Guide: Gen-2 to Gen-3 Spring Batch Applications

## 1. What & Why

This document is a reference and checklist for migrating legacy “Gen-2” Spring Batch applications (Java 8 + Spring Boot 2.x + iBATIS) to “Gen-3” (Java 17 + Spring Boot 3.x + MyBatis). It ensures you follow team standards, upgrade dependencies, refactor code, and use shared templates.

## 2. Step-by-Step Flow

### 2.1 Prepare Your Environment

1. Verify Git Branch:  
 git fetch origin  
 git checkout Dvl-Aws  
 git pull  
\* Ensure the Dvl-Aws branch contains iBATIS→MyBatis changes.  
  
2. Install JDK 17:  
 sudo apt update  
 sudo apt install openjdk-17-jdk  
 java -version # should report Java 17.x  
  
3. IDE Setup (Eclipse):  
 - Project ▶ Properties ▶ Java Build Path ▶ Libraries: Add JRE System Library → JDK 17.  
 - Project ▶ Properties ▶ Java Compiler: Set compliance level to 17.

### 2.2 Grab the Right Gen-3 Template

Your team has two base starter projects:  
- BsPrdTmpltClodBtchDb: DB-writing batch skeleton  
- BsPrdTmpltClodBtchFile: File I/O batch skeleton  
Choose the one matching your workload, copy it locally, and open it in your IDE.

### 2.3 Update Build: Gradle & Wrapper

1. In gradle-wrapper.properties, set:  
 distributionUrl=https://services.gradle.org/distributions/gradle-8.3-bin.zip  
  
2. In build.gradle, update:  
- Spring Boot plugin from 2.x → 3.2.0  
- SourceCompatibility from 1.8 → JavaVersion.VERSION\_17  
- Replace iBATIS dependency with mybatis-spring-boot-starter.

### 2.4 Mirror the Project Structure

Under src/main/java/com/delta/crew/, ensure these packages:  
- btch/ (BatchConfig, tasklets/readers)  
- cnfg/ (MyBatisConfig, filters)  
- cnst/ (constants)  
- dao/ (mapper interfaces or DAO)  
- domn/ (domain/DTO classes)  
- mapper/ (optional XML files)  
- Application.java  
Under src/main/resources/config/, include application-\*.properties and log4j2-\*.properties for each environment.

### 2.5 Wire Up the Batch Job

• For DB jobs, define a single-tasklet step:  
 @Bean public Job myDbJob(...) { ... }  
  
• For File jobs, use chunk-oriented step:  
 @Bean public Job myFileJob(...) { ... }

### 2.6 Configure MyBatis & Mappers

1. Create SqlSessionFactoryBean in MyBatisConfig.  
2. Place XML mappers under resources/mapper or use @Mapper interfaces.  
3. Use @Select and @Update annotations for simple mappings and stored procedures.

### 2.7 Refactor Your DAO

Replace legacy iBATIS code:  
- sqlMapClient.queryForList(...) → mapper.fetchPending()  
- sqlMapClient.update(...) → mapper.callProc(params)

### 2.8 Extras & Cleanup

• Register URL filters via FilterRegistrationBean to skip health endpoints.  
• Ensure package names are com.delta.crew only.  
• Remove any demo code from the template.

### 2.9 Git Workflow

git checkout -b feature/migrate-batch-to-gen3  
... (code changes)  
git commit -m "Migrate to Gen-3"  
git push -u origin feature/migrate-batch-to-gen3

## 3. Questions to Ask Before Starting

- Is Dvl-Aws the correct parent branch? Which template should I use (Db or File)?

- Can you share the DB URLs, credentials, and file paths for DVL/INT/PRD?

- Has all iBATIS→MyBatis migration been done in Dvl-Aws already?

- Does our CI pipeline support Java 17 & Gradle 8.3?

- Are we using Log4j2 or Logback? Where should logs go?

- How should job failures be reported (email, Slack, PagerDuty)?

- What is our cut-over and rollback plan when switching to Gen-3?

- What are the testing requirements (unit/integration tests, coverage)?