spring Batch

Spring Boot Based

배치(Batch) 란?

- 배치는 일괄처리 라는 뜻을 가지고 있습니다.

전 날의 어떤 데이터를 집계한다고 예를 들어보겠습니다.

이 전날의 데이터를 API로 만들어서 실행하면 어떻게 될까요?

🚫 대량의 CPU %, 입/출력 리소스 점유하여 들어오는 요청을 처리하기 힘들어지고,

심각하게는 서버 다운을 초래할 수 있습니다. Ex) 관리자 서버

- 이후 API를 만들어 실행했을 때 중간에 오류가 발생했을 때 그 지점까지의 집계한 모든 데이터는 어떻게 될까요?

- 복구를 했다고 하더라도 저장되다가 말던 오류 지점부터 다시 실행할 수 있을까요?

- 이미 작업된 집계를 어떤 누군가가 한번 더 실행하면 데이터는 어떻게 될까요?

이런 문제점들을 개선하고 대용량의 데이터를 처리하는 애플리케이션을 바로 배치 라고 합니다

배치 애플리케이션

정기 배치	일, 주, 월과 같이 정해진 기간에 정기적으로 수행
이벤트성 배치	특정 조건을 설정해두고 조건이 충족되었을 때만 수행
On-Demand 배치	사용자 요청 시 수행

배치 애플리케이션은 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라

일괄적으로 처리하는 것을 의미합니다.

배치 프로그램은 수행 주기에 따라 구분할 수 있습니다.

배치 프로그램이 갖추어야 하는 필수 요소

1. **대용량 데이터** – 대량의 데이터를 조회, 전달, 계산 등등의 처리가 가능해야 합니다.
2. **자동화** – 심각한 오류를 제외하고는 사용자의 개입이 없이 수행되어야 합니다
3. **견고성** – 잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일이 없어야 합니다.
4. **안정성/신뢰성** – 오류가 발생하면 오류의 발생 지점, 시간 등을 추적할 수 있어야 합니다.
5. **성능** – **다른 애플리케이션을 방해하지 않도록 수행**되고, **지정된 시간 안**에 처리가 완료되어야 합니다.

Spring Batch란

Spring Batch는 일괄 처리를 위한 오픈 소스 프레임워크 입니다.

관리자 서버에서 리소스를 많이 차지하던 스케줄링 푸시 메소드를 이번 기회를 통해 별도의 푸시 서버로 분리를 하게 되었습니다.

Spring Batch는 Spring 프레임워크의 POJO기반 개발 접근 방식을 기반으로 합니다.

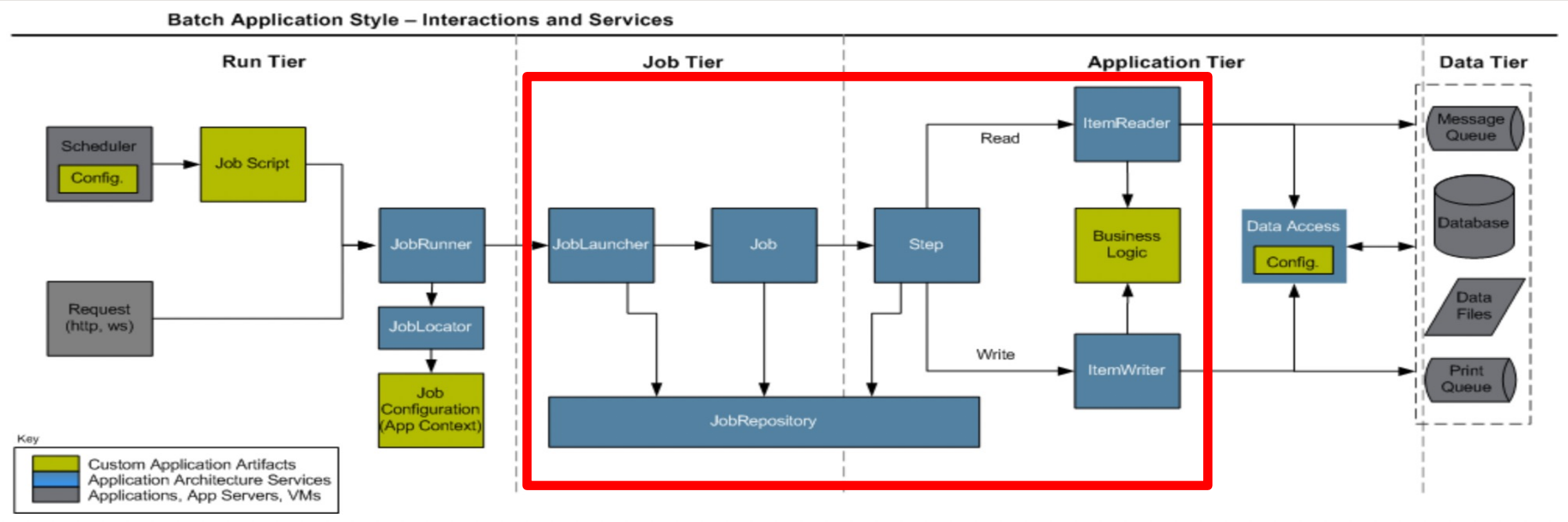
Spring Batch는 로깅/추적, 트랜잭션 관리, 작업 처리 통계, 작업 재시작, 건너뛰기, 리소스 관리 등

대용량 데이터 처리에 필수적인 기능들을 제공해줍니다.

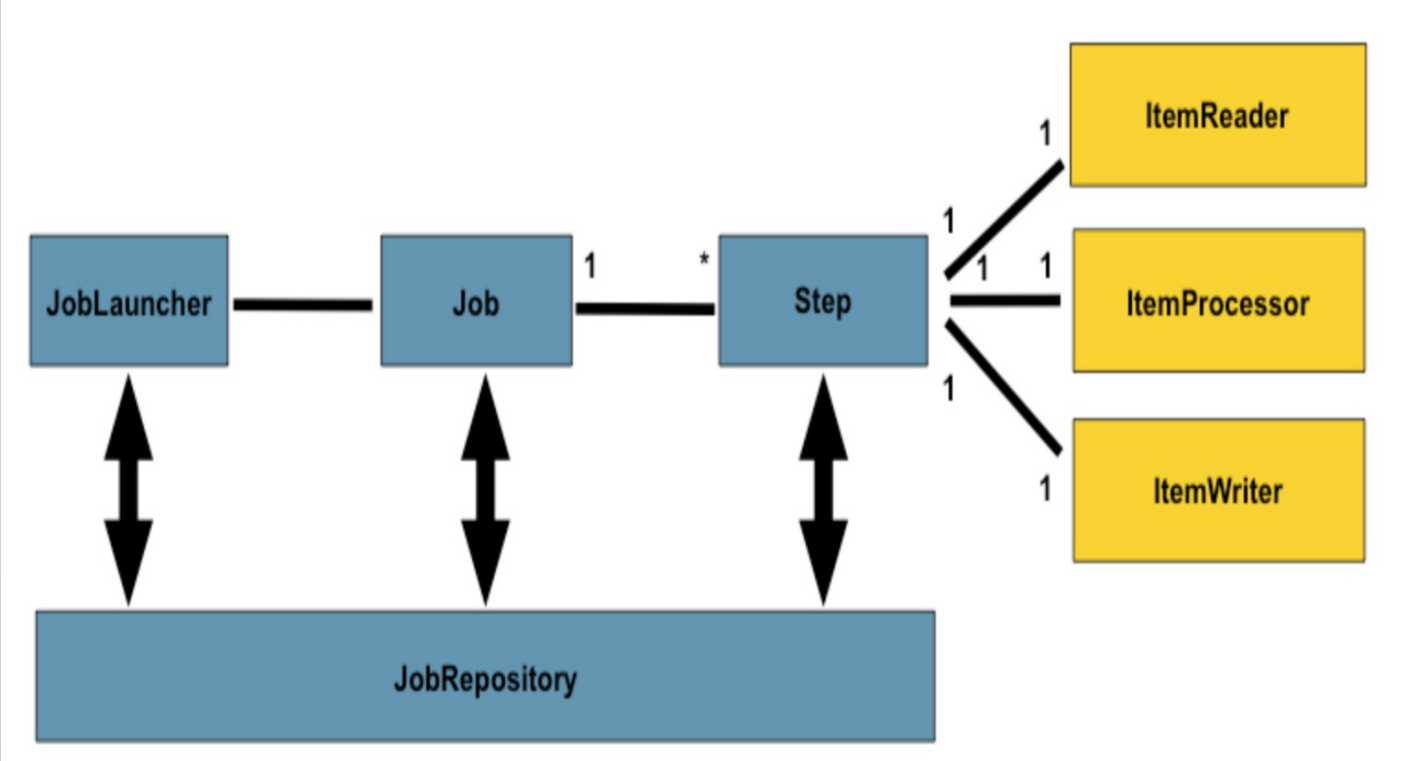
배치와 스케줄러 차이

배치는 일괄처리, 스케줄러는 특정 시간 주기로 설정했던 Batch Job을 자동으로 수행 해주게끔 해주는 도구입니다.

Spring Batch 구조



세부 구조



Spring Batch에서의 Job이란 하나의 배치 작업 단위를 얘기합니다.

Job에는 여러 Step이 존재하고,
Step안에 Tasklet 또는
Item Reader, Processor, Writer가 존재합니다.

Tasklet은 세개로 나눈 구조를 한번에 해줄 수 있는 Step단위입니다.

Reader, Processor, Tasklet 구조의 단위로 나누어서 하나의 Job을 만들 수 는 없습니다 ❌

ItemReader는 필수이며, 이름 그대로 아이템을 읽어들입니다. 그것이 꼭 데이터베이스의 데이터만을 의미하지는 않습니다.

ItemProcessor는 가운데에서 비즈니스 로직을 수행하는데 필수는 아닙니다. 이 단위가 없어도 Chunk 작업은 구성이 가능합니다.

ItemWriter는 필수이며 Processor가 하는일을 이쪽에서 구현해줘도 무방합니다. 여기서의 작업을 명시된 대로 일괄처리를 진행합니다.

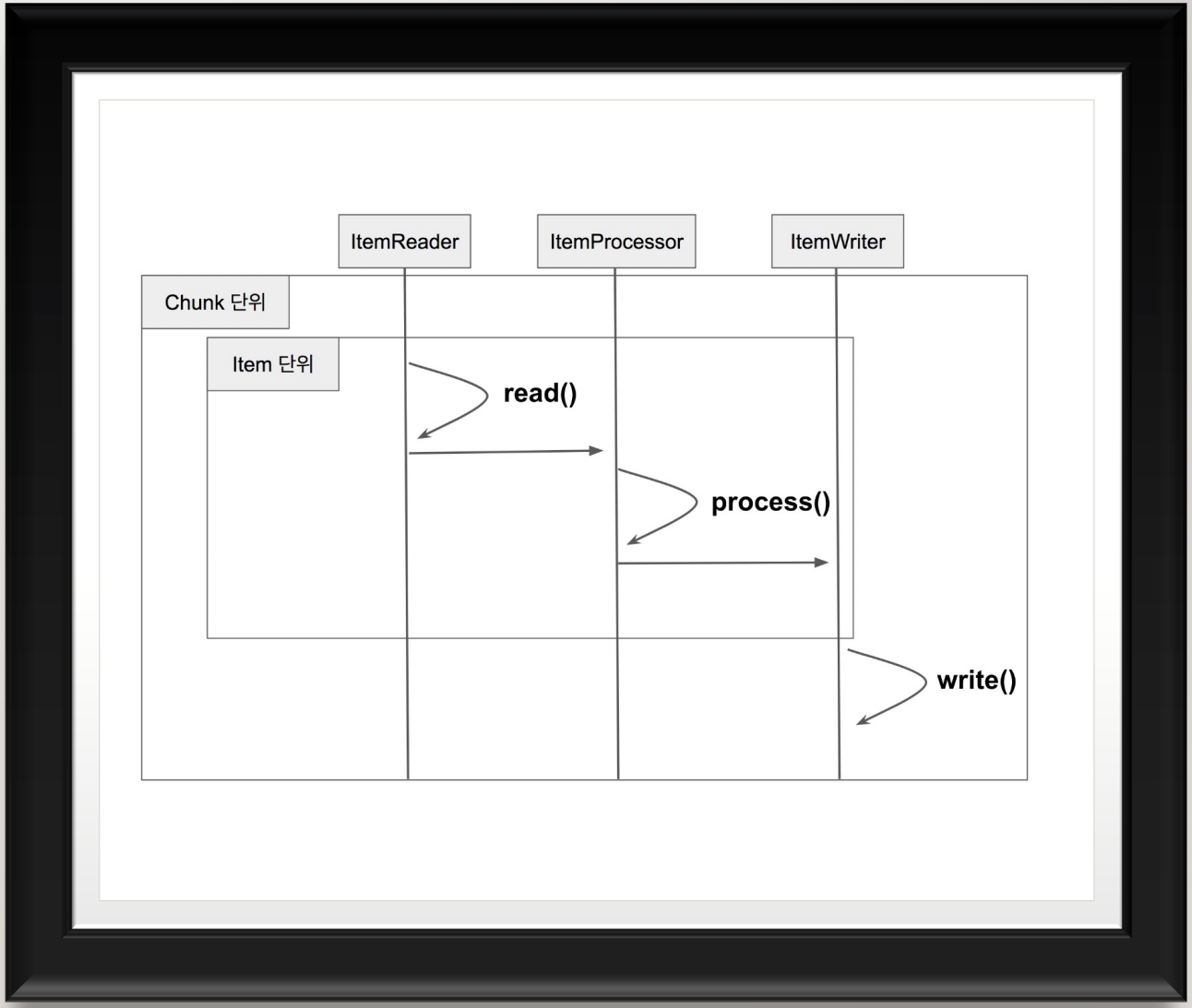
Chunk

Spring Batch의 큰 장점인 Chunk 지향 처리입니다.

Chunk란, 데이터 덩어리로 작업할 때, 각 작업 완료 사이의 처리되는 행의 수를 말합니다.

정리하면, 한번에 하나씩의 데이터를 읽어서 Chunk라는 단위를 만들고, 그 단위로 트랜잭션(데이터베이스의 상태를 변화시키기 위해 수행하는 작업의 단위)을 다루는 것을 의미합니다.

1. Reader에서 1개의 데이터를 가져오고
2. 읽어온 데이터를 Processor에서 가공합니다.
3. 가공된 데이터들을 쌓다가 설정해준 Chunk size가 되면 Writer에 전달하여 Writer가 일괄 저장합니다.



Page size vs Chunk size

많은 양의 데이터를 처리할 때 n개씩 끊어서 조회할 때 페이징 처리를 해주게 되는데,

그 page size와 방금 알아봤던 chunk size는 의미하는 바가 다릅니다.

Chunk size는 한번에 처리될 트랜잭션 단위를 얘기하고,

Page size는 한번에 조회할 Item의 양을 말합니다.

최적으로 배치를 돌려주려면 조회하는 양과 처리되는 양을 일치시켜 주어야 보편적으로 좋은 방법입니다.