대표 프로젝트를 소개해주세요.

프로젝트명	상품 수집/색인 배치 개발
프로젝트기간	2020.01 ~ 2021.02
참여인력	검색TF 개발자3명, QA 1명
사용한 기술스택	ElasticSearch, Elasticsearch Rest High Level Client Java, Spring Boot, Gradle, MyBatis, MySQL

프로젝트 소개 *

홈플러스에서 취급하는 상품에 대해, 검색서비스가 가능하도록 상품에 대한 정보를 수집/색인 합니다.

[색인 설계]

홈플러스의 데이터 구조는 하나의 상품에 대해서도 각각의 점포별로 취급을 해야하기 때문에 색인 구조를 어떻게 잡아야할지 고민이었습니다. ES에 색인할때, Nested구조로 갈지 Flat 구조로 갈지 정해야 했는데, Nested 구조는 상품을 찾고, 다시 내부의 점포별 정보를 찾아야 하므로 비효율적으로 판단되고, 또한 로그인한 사용자는 본인의 위치에 맞는 권역(점포)의 상품정보만 노출하면 되므로 Flat 구조로 결정되었습니다.

[증분 처리]

홈플러스는 점포별 상품의 품절 여부와 정보의 변동이 잦고 데이터양이 많아 정적 수집/색인으로는 시간상으로 처리할 수 없기 때문에 증분 수집/색인이 필요합니다. 상품의 수정날짜 컬럼을 대상으로 증분 대상을 결정하며, 해당 대상을 전부 가져온 뒤, 상품이 품절인지 아닌지를 코드에서 여러 조건으로 비교하여 결정한 후, 색인 Update를 하게 됩니다. 이러한 이유는 상품이 판매 중이었다가 품절 되었을 경우, 증분 수집/색인으로 해당 상품의 상태 값을 품절로 변경하기 위함입니다. 또한 ES에서 자동으로 _id값을 생성하지 않게 하기 위해 _id값을 지정하였는데 상품번호+점포정보로 조합된 String값을 hashcode로 변환하여 저장하는 방식을 선택하였습니다.

[수집/색인 방식]

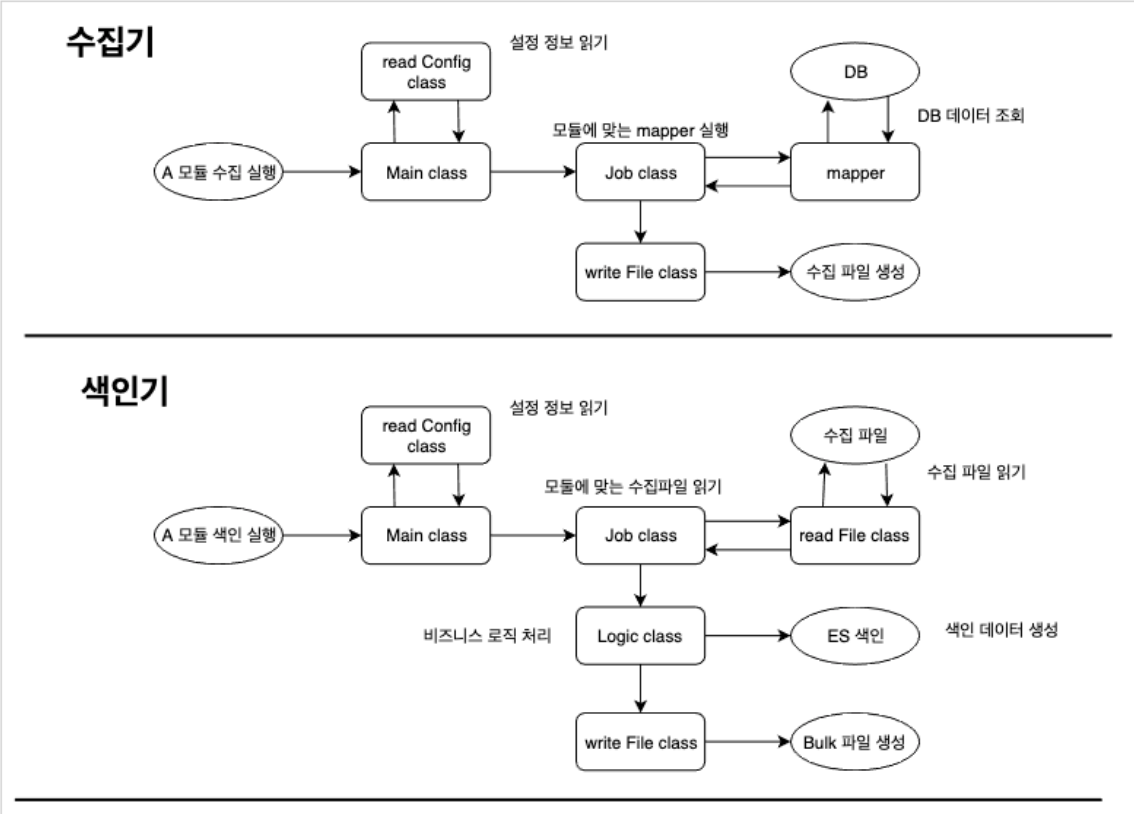
DB에서 Select한 상품의 정보를 바로 색인할지에 대한 여부에 대해서, 전체 수집/색인의 경우 수집이나 색인단계에서 예외가 발생하였을 때, 수집부터 다시 시작해서 기다려야하는 단점이 있습니다. 배치 특성상, 결코 금방 끝날 수 없는 작업이고 이슈 및 예외 발생 시 빠르게 롤백 처리가 되어야 하므로 파일로 Write, Read하는 방식으로 결정하였습니다. 프로젝트를 멀티프로젝트로 만들고 수집과 색인을 독립적으로 수행될 수 있도록 합니다. 수집시 json형태의 파일을 저장하고, 색인시 파일을 읽어서 ES에

bulk로 색인합니다. 색인이 완료된 이후, 색인된 폴더는 backup폴더로 이동합니다. 예외 발생 시에는 프로세스 종료전, error폴더로 이동하며 종료됩니다.

전체 시스템구성 *

1. Jenkins의 timer에 의해 전체 수집/색인 프로세스 동작
2. 수집 시 멀티쓰레드 task별로 json파일 생성
3. 색인 시 [Name_datetime]의 형태로 새로운 색인 인덱스 생성
4. 전체 색인 중 증분색인에 대한 history 기록
5. 새로운 인덱스가 색인되어지는 동안 기존의 인덱스는 서비스 지속(alias값으로 지정되며 형태는 [Name](datetime이 없다)
6. 새로운 색인 인덱스 완료 후 기록된 증분색인도 추가로 완료되면 인덱스 alias 교체
7. 새로운 색인 인덱스로 서비스 시작

[간략한 Work flow]



프로젝트에 기여한 내용 *

[전체 수집 상품 정보 쿼리 작성]
팀원의 기존 작성 쿼리를 이어받아, 상품검색에 필요한 정보를 지속적으로 추가

[증분 대상 상품 번호 쿼리 작성]
테이블 별로 증분대상이 되는 상품들을 가져오는 쿼리를 작성하고 결과를 기록

[행사 할인 정보]
상품에 대해 여러 할인이 적용된 최종 판매 가격을 계산하는 로직 구현. 계산에 필요한 컬럼들은 실제 색인반영에서는 제외

[예외처리]
코드에서 발생할 수 있는 모든 부분에 예외 로깅 처리. CustomException으로 ErrorCode와 메세지를 전달받아 젠킨스에서 예외발생시 빠르게 원인을 파악할 수 있도록 구현

[프로세스 시작 전 상황별 Validation 처리]
파일로 저장하여 처리하는 방식이기 때문에 상황별로 여러 종류의 Validation에 대해서 처리 필요. 각 수집, 색인 프로세스 시작 전, 상황별 체크(예를 들어 수집 중에 예외가 발생하였는데 그대로 색인으로 프로세스가 이어지면 안 되므로 수집폴더의 특정 파일의 유무를 체크하거나 PID의 값으로 수행여부 확인)를 통해서 프로세스의 시작 여부를 결정

[수집속도 향상 방안 - 현재 미적용중]
실제 운영환경에서 택배상품의 경우 상품번호를 조회하는 쿼리의 속도가 매우 느려져서 해결방안 케이스 정리(커버링인덱스, MyBatis ResultHandler, 추가 파일 저장)

프로젝트 성과 *

여러 번의 수정을 거치면서 참여한 해당 프로젝트의 기간은 2020.01 ~ 2021.02까지입니다. (여러 프로젝트도 동일한 기간) 프로젝트에 할당한 시간이 많은 만큼 문제가 타팀에 비해 현저히 적게 발생하였습니다. 레거시 상품정보 수집/색인방식에 비해 3배 이상 빠르게 프로세스가 완료될 수 있었으며, 상품 데이터의 구조도 점포정보, 행사정보 , 상품정보 등의 Object로 나누어 유지보수 시 빠르게 확인 및 처리가 가능하도록 설계하였습니다. 현재는 홈플러스의 정책이 변경되거나 했을 때, 좀 더 빠르게 수집/색인이 이루어질수 있도록 개선방안을 정리 중에 있습니다.

트러블 슈팅 경험 *

[문제정의]
색인단계에서 멀티쓰레드로 범위를 나누어 처리하도록 하였는데, 그 중 일부 쓰레드에서 예외가 발생하였을 경우, 예외 발생 시점에서 바로 프로세스가 종료되지 못하고, 전체 색인 Loop를 행하고 종료되는 문제 발생
[사실 수집]
여러 쓰레드가 동시에 색인데이터를 ES에 전송하니 ES에서 too many request 예외를 발생
[원인추론]
ES에서 권장하는 색인데이터 request의 사이즈는 5Mb인데 색인처리되는 양보다 요청하는 데이터가 더 많아 5Mb이상 넘어서는 요청이 있을 것이라는 추측
[조치 방안]
Block방식으로 메인쓰레드가 요청을 대기하기엔 완료 시간이 너무 길어짐.
설정파일에서 정의한 개수만큼(요청데이터가 5Mb이하가 되도록) Task를 List<CompletableFuture>에 적재하면, block방식으로 CompletableFuture가 모두 get()될때까지 대기하고, 다시 해당 List에 Task를 처리하는 방식으로 처리

방식을 변경하므로써, 부분 block이 발생하였고, 실제 색인 속도도 기존에 비해 저하되었지만 전송데이터에 대한 예외는 더이상 발생하지 않았고, 예외가 발생되는 요청도 Loop만큼 하지 않아 안정성이 보장

아쉬운 점 *

해당 프로젝트를 개발할 때, TDD의 형태로 개발을 하지 않고, 그냥 개발을 진행하였습니다. 그때의 테스트 방식은 직접 여러 번 실제 메소드에 디버깅하여 확인을 하는 것이었습니다. 서비스 오픈 이후, 추가되는 기능 추가 및 변경에 대해서는 Test코드를 작성하여서 처리를 하고 있지만, 테스트 해야 하는 메소드에 다른 로직이 껴있는 부분이 있어 작업에 어려움이 생겼습니다. 프로젝트 리팩토링에 대해서 의견을 제시할 수도 있었겠지만, 정상적으로 서비스 중이며 오랜 기간 이미 테스트를 마친 프로젝트에 다시금 새로이 테스트 하며 진행할 수가 없었습니다. 처음 개발 시에 현재처럼 동료 팀원과 함께 TDD방식까지는 아니더라도, 테스트코드에 대해서 여러 상황으로 개발하였다면, 추후 기능에 대해 테스트할 때 한 메소드에 다른 기능 로직이 섞여 있는 상황은 오지 않을 것이라는 아쉬움이 남게 되었습니다.

참고자료

Github - https://github.com/mertyn88
Blog - https://velog.io/@mertyn88
Portfolio - https://mertyn88.github.io/