강동현 · 백엔드 신입 (Java / Spring)

Email	donghyeonka68@gmail.com
GitHub	github.com/donghyeon-ka
TechLog	hyeonworks.com
Blog	velog.io/@donghyeon2

Summary

Java / Spring Boot 기반 백엔드에서 인증 경계, 데이터 조회 성능, 계층 분리에 집중해 학습해 온 신입 백엔드 개발자입니다.

현업 개발자의 멘토링 아래 약 6개월 동안 과제를 수행하며, 단순히 기술을 적용하는 데서 끝내지 않고 **왜 이 구조를 선택했는지, 어떤 조건에서 문제가 생기는지**를 직접 구현하고 검증했습니다. Keycloak 인증 구조에서는 token / session의 소유 경계를 비교했고, 피드 조회에서는 N+1부터 Top-N 조회와 pagination까지 실제 SQL과 실행계획으로 확인했습니다.

이 과정에서 쌓인 기록을 다시 찾고 맥락대로 연결하기 위해 **TechLog**를 만들고 있습니다. 현재 frontend는 hyeonworks.com에서 확인할 수 있으며, backend와 인프라의 최종 통합은 진행 중입니다.

Skills

Core

- Java 21, Spring Boot 4, Spring Security, Spring Data JPA, PostgreSQL

Architecture / Testing

- Clean Architecture, Gradle multi-module, ArchUnit, JUnit, Testcontainers, Flyway

Auth / Performance

- Keycloak, OAuth2 / OIDC, PKCE, JWT / JWKS, Redis, Hibernate Statistics, `EXPLAIN (ANALYZE, BUFFERS)`

Projects

TechLog — 기술 학습 기록을 서비스로 연결한 프로젝트

> 멘토링 기반 개인 프로젝트 · 약 6개월 · 1 인 · 설계 / 구현 / 검증 hyeonworks.com

인증, 성능, 아키텍처 과제를 진행하며 문서가 늘어났지만, 기록이 흩어져 **"왜 이런 결정을 했는지"를 다시 따라가기 어려운 문제**가 생겼습니다. 이를 해결하기 위해 Public / Studio 구조의 기술 기록 서비스를 만들었습니다.

- 작성 흐름을 Create / Save / Validate / Preview / Publish 유스케이스로 분리하고 persistence / rendering 구현을 port 뒤로 분리했습니다.
- backend를 18개 Gradle module로 나누고, `modules.json`을 기준으로 허용 dependency와 runtime membership을 검증하도록 구성했습니다.
- Project / Topic / Release 관리 유스케이스를 application layer에 두어 JPA entity와 화면 DTO가 application 정책으로 직접 유출되지 않도록 분리했습니다.
- backend contract에는 값이 있지만 HTTP mapping 또는 frontend adapter에서 누락되어 화면이 비는 문제를 추적하고 contract / route / test gate를 보강했습니다.
- frontend도 `domain → application → adapter → presentation` 경계를 두고 HTTP / mock adapter를 교체할 수 있도록 구성했습니다.

`Java 21` `Spring Boot` `JPA` `PostgreSQL` `Gradle` `ArchUnit` `React` `TypeScript`

Keycloak 인증 구조 — 브라우저와 서버의 credential 경계 비교

> 멘토링 과제 · 개인 lab · 설계 / 구현 / 검증

같은 로그인 기능이라도 SPA, BFF, Forward-Auth에 따라 token과 session의 위치, API 호출 주체가 달라집니다. 구조 이름만 비교하지 않고 **credential이 실제로 어디에 머무는지**를 기준으로 네 가지 패턴을 구현했습니다.

- SPA / Token Mediator / BFF / Forward-Auth를 구현해 authorization code 교환 주체, refresh token 보관 위치, API caller를 비교했습니다.
- BFF에서는 브라우저 JavaScript에서 OAuth token을 제거하는 대신 server-side session, CSRF, session scale-out 문제가 새로 생기는 것을 확인했습니다.
- Forward-Auth에서는 edge가 만든 identity header를 신뢰하려면 header overwrite와 network isolation이 함께 필요하다는 점을 검증했습니다.
- 이후 2-node K3s 환경으로 확장해 Keycloak session의 실제 저장 위치를 확인했습니다.
- Keycloak 26 기본 `persistent-user-sessions` 조건에서 교차 노드 session 처리가 Infinispan 복제보다 **공유 PostgreSQL session 조회에 의존하는 경로**임을 SQL로 확인했습니다.

`Keycloak` `OAuth2 / OIDC` `Spring Security` `PostgreSQL` `K3s` `Nginx`

피드 조회 성능 — N+1 이후의 조회 전략 비교

> 멘토링 과제 · PostgreSQL 16 Testcontainers 기반 측정 아래 수치는 local / warm-cache 기준이며 운영 성능 수치가 아닙니다.

피드마다 최신 하이라이트와 공개 범위, pagination을 함께 처리하면서 N+1만 제거하면 끝나지 않는 문제를 확인했습니다. 쿼리 수, 반환 행, 인덱스, paging 비용을 분리해 비교했습니다.

- FeedItem 1,000개 조회에서 Hibernate Statistics 기준 PreparedStatement **2,022개**를 기준선으로 확인했습니다.
- fetch join에서 `MultipleBagFetchException`, row 증가, collection fetch + paging의 in-memory paging 문제를 재현했습니다.
- Batch Fetch 적용 후 같은 조건에서 PreparedStatement를 **2,022 → 23**으로 줄였습니다.
- 부모별 최신 3개 조회에서 Window Function / LATERAL / 2-step을 비교하고, 작은 K와 복합 인덱스 조건에서 LATERAL이 유리한 이유를 실행계획으로 확인했습니다.
- 2,000행의 깊은 페이지에서 OFFSET은 2,000행을 읽고 Keyset + 정렬키 index는 **20행**만 읽는 실행계획을 확인했습니다.

`Spring Data JPA` `Hibernate` `PostgreSQL 16` `Testcontainers` `EXPLAIN ANALYZE`

Links

- **TechLog**: hyeonworks.com
- **GitHub**: github.com/donghyeon-ka
- **Blog**: velog.io/@donghyeon2