강동현 · 백엔드 신입 (Java / Spring)

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

Summary

Java / Spring Boot 기반 백엔드에서 인증, 계층 분리, 예외 응답 표준화, 데이터 영속성 경계에 집중해 학습해 온 신입 백엔드 개발자입니다.

Spring Boot 인증 서버를 5 개 모듈 Clean / Hexagonal 구조로 나누고, `domain` 모듈이 Spring / JPA / Servlet에 의존하지 않도록 분리했습니다. 또한 ArchUnit 테스트로 계층 의존성을 검증하고, Keycloak JWT를 Resource Server로 다시 검증하는 구조를 구현했습니다.

프로젝트는 개인 로컬 / dev 환경 기준입니다. 대신 구현 범위, 검증 범위, 아직 운영 경험으로 볼 수 없는 부분을 문서 안에서 구분했습니다.

Skills

Core

- Java 21, Spring Boot 4, Spring Security Resource Server, Spring Data JPA, PostgreSQL 16

Architecture / Testing

- Clean Architecture, Hexagonal Architecture, Gradle multi-project, ArchUnit, JUnit 5, AssertJ

Auth / Observability

- Keycloak, OAuth2 / OIDC, JWKS 기반 JWT 검증, Flyway, OpenAPI / Swagger UI, MDC `traceId` 기반 structured logging

Projects

Project-Auth-Server — Spring Boot 인증 서버

> 개인 프로젝트 · 2025.11 ~ 현재 · 1 인 · 설계 / 구현 / 로컬 검증 github.com/donghyeon-ka/project-auth-server

Spring Boot 인증 서버를 계층별 책임이 드러나도록 분리하고, 인증 위임 / JWT 검증 / 예외 응답 / DB 마이그레이션 경계를 정리한 프로젝트입니다.

- Gradle multi-project로 `domain` / `application` / `presentation` / `infrastructure` / `bootstrap` 5 개 모듈을 분리했습니다.
- `domain` 모듈은 Spring, JPA, Servlet 의존성을 두지 않고 값 객체와 도메인 규칙만 포함하도록 구성했습니다.
- ArchUnit의 `LayerDependencyArchitectureTest`로 `presentation → infrastructure` 같은 계층 위반을 빌드 단계에서 탐지하도록 했습니다.
- 로그인 플로우는 Keycloak / Ingress 계층에 위임하고, 백엔드는 `oauth2-resource-server`로 JWT issuer / signature / expiration / role claim을 다시 검증하도록 구성했습니다.
- `ErrorCode`는 비즈니스 의미만 담고, HTTP status 매핑은 presentation의 mapper가 담당하도록 분리했습니다.
- 예외 처리는 Security filter, Spring MVC 입력 검증, application, infrastructure 누수 감지로 나눴고, 응답은 `ApiResult<T>` 형태로 통일했습니다.
- `TraceIdFilter`로 32 자리 hex `traceId`를 응답 헤더, 로그, audit 흐름에 동일하게 전파했습니다.
- Flyway V1~V5로 `auth` 스키마와 사용자 / provider 관련 변경을 단계별 마이그레이션으로 관리했습니다.

`Java 21` `Spring Boot` `Spring Security` `Keycloak` `JPA` `PostgreSQL` `ArchUnit` `Flyway`

feed-postgresql-query-tuning — PostgreSQL 쿼리 최적화

> 개인 프로젝트 · 2025.03 ~ 2025.04 · 1 인 · 분석 / 최적화 / 검증 github.com/donghyeon-ka/feed-postgresql-query-tuning

피드 목록 조회 쿼리를 실행 계획 기준으로 분석하고, JOIN 구조와 인덱스 설계를 바꿔 성능을 개선한 프로젝트입니다.

- `EXPLAIN ANALYZE` 기준 피드 목록 조회 쿼리 실행 시간을 **1.77 s → 7.66 ms**로 개선했습니다.
- 4-table `LEFT JOIN`에 따른 row 폭증, 인덱스 미사용, 디스크 기반 외부 정렬을 병목으로 확인했습니다.
- `EXISTS` 서브쿼리, 페이지네이션 쿼리 분리, 복합 / 커버링 인덱스를 단계적으로 적용했습니다.
- 단일 인스턴스 환경에서 Locust 1,000 동시 사용자 시나리오를 실행해 약 **900~1,100 RPS** 구간을 확인했습니다.

`PostgreSQL` `EXPLAIN ANALYZE` `Index design` `JPA` `Redis` `Locust`

Project-Infra — Spring 앱 실행 환경

> 개인 프로젝트 · 2025.11 ~ 현재 · 1 인 · 로컬 / dev 환경 구성 github.com/donghyeon-ka/project-infra

auth-server가 동작하는 로컬 / dev K3s 환경을 구성하면서 백엔드와 인프라의 인증 책임을 분리한 프로젝트입니다.

- Traefik ForwardAuth + oauth2-proxy + Keycloak으로 로그인 / 세션 확인을 Ingress 계층에 두고, auth-server는 JWT 검증에 집중하도록 구성했습니다.
- Vault → VSO → Kubernetes Secret 흐름으로 secret 주입 책임을 애플리케이션 코드 밖으로 분리했습니다.
- Kustomize base / overlay 구조로 로컬과 dev 환경 차이를 manifest 단에서 관리했습니다.

`Kubernetes` `K3s` `Kustomize` `Vault` `VSO` `Traefik`

Links

- **GitHub**: github.com/donghyeon-ka
- **메인 프로젝트**: project-auth-server
- **인프라**: project-infra
- **쿼리 튜닝**: feed-postgresql-query-tuning
- **Blog**: velog.io/@donghyeon2