# **Spring Boot 학습 로드맵 (Node/Nest 경험자용)**

**대상:** Express/NestJS/React/Arduino를 초급 이상으로 쓰는 자바 초심자

**목표:** 6–8주 안에 Spring Boot로 “테스트가 있는 실무 수준의 REST API 서비스”를 만들고 배포까지 경험하기

## **0. 멘탈 모델: NestJS ↔ Spring Boot 빠른 대응표**

| **개념(당신이 아는 것)** | **NestJS** | **Spring Boot** |
| --- | --- | --- |
| 애플리케이션 부트스트랩 | NestFactory.create | @SpringBootApplication + SpringApplication.run |
| DI 컨테이너 | @Injectable + 모듈 | @Component/@Service/@Repository + 컴포넌트 스캔 |
| 모듈 | @Module | 패키지 단위 + @Configuration, @Bean |
| 컨트롤러 | @Controller/@Get/@Post | @RestController/@GetMapping/@PostMapping |
| 파이프/DTO 검증 | class-validator | Bean Validation(@Valid, @NotNull 등) |
| 예외 처리 | @Catch + 필터 | @ControllerAdvice + @ExceptionHandler |
| 미들웨어 | Middleware | 서블릿 **Filter**, Spring **HandlerInterceptor** |
| 가드/권한 | Guards | Spring Security 필터 체인 + 메서드 시큐리티(@PreAuthorize) |
| 설정/환경 | ConfigModule | application.yml + @ConfigurationProperties + Profiles |
| ORM | TypeORM/Prisma | JPA(Hibernate) + Spring Data JPA |
| 테스팅 | Jest | JUnit 5 + Spring Test + Testcontainers |

**익숙해질 포인트**: 어노테이션 기반 설정과 **빈(Bean)** 라이프사이클, **트랜잭션** 개념이 핵심입니다.

## **1. 환경 설정 체크리스트 (1일)**

* **JDK 21**(권장) 설치 (SDKMAN/ASDF 등 버전관리 툴 추천)
* **IntelliJ IDEA** 설치 → 프로젝트 JDK 지정, Gradle 사용 권장
* **Gradle** 또는 Maven: 이 로드맵은 **Gradle** 기준
* 로컬 DB: Docker로 **PostgreSQL** 또는 **MySQL** 준비
* API 점검 도구: IntelliJ HTTP Client 또는 Postman/Insomnia
* 프로젝트 생성: **Spring Initializr**에서 다음 기본 의존성 선택
  + Spring Web, Validation, Spring Data JPA, (DB 드라이버), Lombok(선택), Spring Boot DevTools(선택)

**미션**: “hello-springboot” 프로젝트를 생성하여 /health 엔드포인트(200 OK)까지 띄우기.

## **2. 자바 필수 문법(스프링 입문용) (3–5일)**

스프링에 필요한 **실전 자바만** 골라 학습합니다.

* 클래스/객체, 접근제어자, 상속 vs 컴포지션, **인터페이스**
* **제네릭**, **컬렉션(List/Map)**, **Optional**
* **예외 처리**(체크/언체크), try-with-resources
* **레코드(record)**(DTO 용도), **람다 & 스트림**(기초)
* 빌드/실행 사이클: Gradle 태스크(bootRun, test, build)
* 단위 테스트: **JUnit 5**, **AssertJ** 기초

**연습**: Money, Email, UserId 같은 **값 객체(Value Object)** 를 record로 구현하고, 유효성 검증 테스트 작성.

## **3. Spring Boot 코어 (3–4일)**

* **애플리케이션 구조**: 패키지 설계(계층형 or 도메인형)
* **빈과 DI**: @Component/@Service/@Repository, **생성자 주입** 고수
* **설정**: application.yml, **Profiles**(dev, test, prod), @ConfigurationProperties
* **로깅**: Logback 기본, 로깅 레벨 제어
* **유효성 검증**: @Valid, @NotNull 등 Bean Validation + 예외 메시지 표준화

**연습**: @ConfigurationProperties로 app.\* 설정을 주입받아 /config에서 노출(비밀 값 제외).

## **4. 웹 레이어: Spring MVC (1주)**

* **컨트롤러**: @RestController, @RequestMapping, 바인딩(@PathVariable, @RequestParam, @RequestBody)
* **DTO & 매퍼**: 요청/응답 DTO 분리, MapStruct(선택)
* **에러 처리**: 전역 예외 처리기 @ControllerAdvice + 표준 에러 포맷
* **필터 vs 인터셉터**: 로깅/추적 ID, CORS 처리 위치 이해
* 파일 업로드, 페이징/정렬(Pageable)

**연습**: todos REST API (CRUD) + 전역 에러 포맷 + 간단한 요청 로깅 인터셉터.

## **5. 데이터 접근: JPA & 트랜잭션 (1–1.5주)**

* JPA 엔티티 매핑: 식별자, 값 타입, 연관관계(1:N, N:1, N:M 지양), 영속성 컨텍스트
* **Spring Data JPA**: Repository 인터페이스, 메서드 쿼리, @Query
* 트랜잭션: @Transactional 전파/격리 기초, **지연 로딩**과 N+1 이슈
* 쿼리 최적화: 페치 조인, 페이징 전략
* DB 마이그레이션: **Flyway**(권장) 또는 Liquibase

**연습**: 게시판(Posts, Comments) 모델링 → 연관관계 매핑 → CRUD + 페이지네이션 + 마이그레이션 스크립트.

## **6. 테스트 전략 (동행 학습, 3–5일 집중)**

* 계층별 테스트: 단위, 슬라이스, 통합
* **Spring Test**: @WebMvcTest, @DataJpaTest, @SpringBootTest
* **MockMvc**로 컨트롤러 테스트, **Testcontainers**로 실DB 통합 테스트
* 픽스처/빌더, 테스트 데이터 전략

**연습**: users API에 대해 컨트롤러 슬라이스 테스트와 JPA 슬라이스 테스트 각각 작성.

## **7. 보안: Spring Security (1주)**

* 보안 필터 체인, SecurityFilterChain 구성
* 기본 로그인/패스워드 인코딩, **JWT** 기반 로그인(토큰 발급/검증)
* 역할/권한, 메서드 보안(@PreAuthorize)
* CORS, CSRF 전략 이해

**연습**: /admin/\*\*는 ROLE\_ADMIN만, /me는 자기 자신만 접근 가능하도록 구성.

## **8. 운영 준비: 관측/배포 (3–5일)**

* **Actuator**로 헬스체크/메트릭 노출
* **Micrometer** + 로그 구조화(추적ID), 에러 로깅 표준
* 이미지 빌드: **Buildpacks(bootBuildImage)** 또는 Dockerfile
* 외부 설정: 환경변수/ConfigMap, 프로파일 분리
* 간단 배포: Docker Compose 로컬, 또는 Render/Railway/AWS Elastic Beanstalk 등

**연습**: prod 프로파일로 실행 시 DB 연결/로깅 레벨이 달라지도록 구성.

## **9. 선택: Reactive(WebFlux)**

* 쓰레드 모델 차이(MVC vs Netty), Mono/Flux 기초
* 블로킹 라이브러리와의 궁합 주의

**연습**: SSE로 알림 스트림 만들기.

## **10. 캡스톤 미니 프로젝트 제안 (난이도 순)**

1. **URL Shortener**

* 해시 생성, 충돌 처리, 리다이렉트, 클릭 카운트(캐시)
* 학습 포인트: MVC, JPA, 예외처리, 캐시

1. **투두+코멘트 REST 서비스**

* 사용자 인증(JWT), 권한, 페이지네이션
* 학습 포인트: Security, DB 마이그레이션, 테스트

1. **IoT 텔레메트리 수집(Arduino 연동)**

* 장치 등록/키 발급, 측정치 수집 API, 경보 규칙, 최근 데이터 조회
* 학습 포인트: 설계, 성능, 관측, 배포

각 프로젝트는 **테스트 우선**(컨트롤러 슬라이스 + JPA 슬라이스 + 통합)으로 진행하세요.

## **11. 하루 공부 루틴(90분 샘플)**

1. 개념 20분(공식 문서/가이드 읽기)
2. 실습 45분(오늘의 미션 구현)
3. 테스트 15분(레드→그린)
4. 회고 10분(배운 점/막힌 점 기록)

## **12. 체크리스트 & 자가 퀴즈**

**스스로에게 물어보기**

* 스프링의 DI 컨테이너가 어떤 타이밍에 어떤 객체를 만들고 주입하는가?
* 트랜잭션 경계 밖에서 Lazy 로딩이 터지는 이유는?
* 예외를 어디서 잡아야 비즈니스 규칙과 인프라 오류가 분리되는가?

## **13. 자주 막히는 포인트(노드 개발자 관점)**

* 동시성/쓰레드: MVC는 요청당 쓰레드. 블로킹 I/O 사용 시 별도 고려 X(일반적 규모). WebFlux는 다름.
* 빌드 산출물: 자바는 **JAR** 실행, 클래스패스 개념 숙지.
* 런타임/컴파일타임 애노테이션: Lombok은 컴파일 타임 변환. 과의존 주의.
* 예외: 체크예외가 존재. 레이어드 아키텍처 별 예외 정책 정하기.