0. 목차

1. 개요 및 개발 환경 설정

- Spring 프레임워크의 역사와 철학
- Spring vs Spring Boot vs Spring Cloud
- Java EE와의 차이점
- 프로젝트 생성 방식
  - Spring Initializr
  - Spring CLI
- 빌드 도구 선택
  - Maven 설정
  - Gradle 설정
- 개발 환경 구성
  - IntelliJ, Eclipse, VS Code
  - Lombok 플러그인 설치
- Spring Boot 버전 전략
  - SNAPSHOT, M, RC, RELEASE
- 디렉터리 구조 설명
  - `src/main/java`
  - `src/main/resources`
  - `application.properties` / `application.yml`

2. 전체 계층 구조 개요 (Layered Architecture)

- 계층형 아키텍처란?
- 전통적 5계층 구조의 역할과 경계
- 각 계층 간 의존성 방향과 책임
- 도메인 주도 설계(Domain-Driven Design) 관점에서 본 계층 구조

3. 프레젠테이션 계층 (Presentation Layer)

- 컨트롤러 역할과 책임
- `@Controller`, `@RestController` 차이
- View 렌더링 vs API 응답 방식
- 클라이언트 요청 매핑: `@RequestMapping`, `@GetMapping`, `@PostMapping`
- 요청/응답 변환: `@RequestBody`, `@ResponseBody`, JSON 직렬화
- 입력 검증 및 오류 처리: `@Valid`, `BindingResult`, `@ExceptionHandler`

4. 애플리케이션 계층 (Application Layer) — 선택적

- 복잡한 유스케이스 처리 (Command, Use Case 중심)
- 도메인 로직 조합 및 트랜잭션 조율
- 애플리케이션 서비스 vs 도메인 서비스
- 예: `UserRegistrationService`, `OrderApplicationService`
- DTO/Command 객체 패턴

5. 서비스 계층 (Service Layer)

- `@Service` 클래스의 역할
- 트랜잭션 범위 지정: `@Transactional`
- 다수의 Repository 및 외부 API 통합 처리
- 서비스 계층에서의 유효성 검사, 비즈니스 규칙 적용
- 예: `UserService`, `OrderService`, `PaymentService`

6. 도메인 계층 (Domain Layer)

- 도메인 모델과 엔티티의 정의
- `@Entity`, `@Id`, `@ValueObject` 개념
- 도메인 이벤트, 도메인 서비스
- 도메인 주도 설계의 원칙
- 애그리거트 루트, 엔티티, VO(Value Object)
- 순수 자바 객체 설계

7. 데이터 접근 계층 (Spring Data)

- JDBC 연결
- JdbcTemplate
- MyBatis 연동
- ORM 개념
- JPA와 Hibernate
- Entity 매핑
  - `@Entity`, `@Table`, `@Id`, `@GeneratedValue`, `@Column`
- Spring Data JPA
  - `JpaRepository`, `CrudRepository`
  - `@Query`, Query Method
- 페이징과 정렬
- Fetch 전략
  - Lazy vs Eager

- 영속성 컨텍스트
- 트랜잭션 처리
  - `@Transactional`
- N+1 문제와 해결 전략
  - Fetch Join
  - EntityGraph
  - Batch Size

8. IoC와 DI (Inversion of Control / Dependency Injection)

- IoC 컨테이너 개념
- BeanFactory vs ApplicationContext
- Bean 등록 방식
  - XML 기반 등록
  - 자바 기반 등록 (`@Bean`, `@Configuration`)
  - 컴포넌트 스캔 (`@Component`, `@Service`, `@Repository`, `@Controller`)
- 의존성 주입 방식
  - 생성자 주입
  - 세터 주입
  - 필드 주입
- 빈 스코프
  - singleton, prototype, request, session, application
- 빈 라이프사이클
  - 초기화, 소멸
  - `@PostConstruct`, `@PreDestroy`
- 조건부 Bean 등록 (`@Conditional`, `@Profile`)

9. AOP (Aspect-Oriented Programming)

- AOP 개념과 필요성
- 핵심 용어: JoinPoint, Pointcut, Advice, Aspect
- `@Aspect` 설정 방식
  - `@Before`, `@After`, `@Around`, `@AfterReturning`, `@AfterThrowing`
- AOP 프록시 구현 방식
  - JDK Dynamic Proxy
  - CGLIB Proxy
- 실전 활용 예
  - 로깅
  - 보안

    - 트랜잭션

10. 웹 계층 (Spring MVC) 심화

- DispatcherServlet 구조
- 요청 매핑
    - `@RequestMapping`, `@GetMapping`, `@PostMapping`
- 요청 파라미터 처리
    - `@RequestParam`, `@PathVariable`, `@ModelAttribute`, `@RequestBody`
- 응답 처리
    - `@ResponseBody`
    - JSON 직렬화
- View Resolver
    - Thymeleaf, JSP
- 유효성 검사
    - `@Valid`, `@Validated`
    - `BindingResult`
- 예외 처리
    - `@ExceptionHandler`, `@ControllerAdvice`
- 파일 업로드 처리
- 국제화(i18n)

11. 테스트와 품질 관리

- 단위 테스트와 통합 테스트
- JUnit 5 설정
- Spring TestContext Framework
- `@SpringBootTest`, `@WebMvcTest`, `@DataJpaTest`
- Mockito와 `@MockBean`
- 테스트 더블
- 테스트 DB 설정
- 테스트 데이터 삽입 (`@Sql`)
- 테스트 자동화
- 커버리지 측정 도구
    - JaCoCo, SonarQube

12. 보안 (Spring Security)

- Spring Security 구조와 필터 체인
- Form 로그인 처리

- 사용자 인증
- 사용자 권한 처리
  - `@Secured`, `@PreAuthorize`, `@PostAuthorize`
- 커스텀 인증 및 인가
- 암호화
  - PasswordEncoder, BCrypt
- 세션 관리
- Remember-me 기능
- CSRF 보호
- CORS 설정
- JWT 기반 인증
- OAuth2 클라이언트 및 서버

13. Spring Boot 심화

- Spring Boot 자동 설정 원리
- `@SpringBootApplication`
- `@EnableAutoConfiguration`, `@Conditional*` 계열 어노테이션
- 외부 설정 바인딩
  - `@ConfigurationProperties`, `@Value`
- 다중 설정 파일 관리
- 프로파일 기반 설정
  - `@Profile`
- 커스텀 Starter 생성
- DevTools, Hot Reload

14. REST API 설계 및 문서화

- RESTful API 설계 원칙
- 상태 코드 정의
- REST 예외 처리 전략
- 표준 응답 구조 설계
- API 버전 관리 전략
- Swagger UI 설정
- SpringDoc OpenAPI 설정
- REST Docs

15. 비동기 처리 및 이벤트 아키텍처

- 비동기 처리 (`@Async`)

- Executor 설정
- 이벤트 발행 및 구독
  - `ApplicationEvent`, `@EventListener`
- 도메인 이벤트 패턴
- 트랜잭션 이벤트 처리

16. 메시징과 통합

- Kafka 연동
  - Producer, Consumer
  - `@KafkaListener`
- RabbitMQ 연동
  - `@RabbitListener`
- 메시지 컨버터
- 메시지 재처리와 리트라이 전략
- Dead Letter Queue (DLQ)
- Spring Integration 기본 구조

17. 마이크로서비스 아키텍처 (Spring Cloud)

- 서비스 디스커버리: Eureka
- 구성 서버: Spring Cloud Config
- API Gateway: Spring Cloud Gateway
- 로드 밸런싱: Spring Cloud LoadBalancer
- 장애 대응: Resilience4j
- 트레이싱: Sleuth, Zipkin
- 이벤트 기반 연동: Spring Cloud Stream
- 인증과 보안: OAuth2 Resource Server
- 분산 환경에서의 트랜잭션 처리 (Saga, Eventual Consistency)

18. 캐시 및 성능 최적화

- Spring Cache 추상화
  - `@Cacheable`, `@CacheEvict`, `@CachePut`
- 캐시 저장소: Redis, Caffeine
- TTL, LRU 전략 설정
- 동시성 고려한 캐시 업데이트
- API 응답 캐싱

19. 운영 및 배포 환경

- Spring Boot Actuator
- Prometheus, Grafana 모니터링 연동
- 환경 분리 설정
  - `application-dev.yml`, `application-prod.yml`
- 로깅 설정: Logback, Log4j2
- 로그 수집기 연동: ELK, Loki
- Docker 기반 배포
- Kubernetes 배포
- Helm Chart
- CI/CD 구성 (GitHub Actions, Jenkins, GitLab CI)

20. 고급 내부 구조 및 커스터마이징

- Spring 컨테이너 동작 방식
- 빈 등록 과정
- `BeanDefinition`, `BeanFactoryPostProcessor`, `BeanPostProcessor`
- 클래스패스 스캔 원리
- 커스텀 어노테이션과 메타 어노테이션
- SPI 구조
- ClassLoader 및 Reflection 활용

21. 실전 프로젝트 설계 예시

- 게시판 및 댓글 시스템
- 쇼핑몰 주문 시스템
- 회원가입 및 JWT 인증 시스템
- Kafka 기반 알림 시스템
- 파일 업로드 및 썸네일 처리 시스템
- 마이크로서비스 기반 도서 API 시스템