2023.7.07 하루일

제 10장 Spring AOP의 개념 이해

AOP (Aspect-Oriented Programming)

- 관점 지향 프로그래밍
- 트랜잭션 처리, 로깅, 보안 등의 공통된 기능 (관심사, Concern)을 비즈니스 로직에서 분리하여 한 곳에 관리하는 기법.
- AOP의 핵심 개념
  - Aspect (관점): 여러 객체에서 공통으로 사용하는 기능을 모듈화한 단위. (예: 트랜잭션, 로깅)
  - Join Point (조인 포인트): Aspect를 적용할 수 있는 실행 지점.
    Spring AOP에서는 메서드 실행 시점을 의미
  - Advice (어드바이스): 실제로 실행되는 공통 기능의 코드
    언제, 어떤 동작을 수행할지 정의
  - Pointcut (포인트 컷): 어떤 Join Point에 Advice를 적용할지 지정하는 조건식 또는 표현식.
  - Weaving (위빙): Advice와 PointCut을 실제 대상 객체에 결합 (적용)하는 과정

Spring AOP의 구현 방식: 프록시 기반 런타임 위빙

- 프록시 (Proxy) 기반: 스프링은 AOP를 적용하기 위해 프록시 객체 (대리 객체)를 사용
- 동작 흐름: 클라이언트가 원본 객체 대신 프록시 객체를 호출
  프록시는 실제 로직 (Target) 실행 전후에 부가 기능을 끼워 넣습니다.
- Weaving 시점: 스프링 AOP의 기본 방식은 런타임 시점에 프록시를 생성하여 결합하는 것

@Transactional 및 @Vaild 연관성

- @Transactional: 트랜잭션 시작/커밋/롤백을 AOP 프록시를 통해 자동으로 처리
  프록시가 호출을 감지하여 트랜잭션을 시작하고, 예외 발생 시 롤백, 정상 종료 시 커밋을 수행
- @Valid: AOP의 @Before Advice처럼 메서드 호출 직전 (Join Point)에 동작하여 입력 데이터의 유효성을 검사
  검증에 실패하면 메서드 본문은 실행되지 않는다.

제11장 AI로 알아보는 트랜잭션과 Validation

트랜잭션 관리 (@Transactional)

- 개념: DB에서 여러 작업을 하나의 단위로 묶어 처리하여, 데이터의 일관성과 안전성을 보장
- Spring 적용: @Transactional 어노테이션을 메서드나 클래스에 붙여 선언적 트랜잭션 관리를 수행
  스프링은 AOP(프록시)를 이용해 commit()과 rollback()을 자동으로 제어

입력 값 검증 (Validation)

- 필요성: 잘못된 값이 DB에 저장되는 것(데이터 무결성)을 방지. 시스템 안정성 유지. 보안 강화
- 종류: 브라우저에서 실행되는 클라이언트 측 검증, 서버에서 실행되는 서버 측 검증
- Bean Validation 활용: javax.validation or jakarta.validation 패키지의 @NotNull, @Email, @Size, @Min 등의 어노테이션을 DTO에 사용하여 검증 규칙을 정의

@Valid와 BindingResult를 통한 처리 과정

- 검증 수행: 컨트롤러 메서드에서 @Valid를 DTO에 적용하여 유효성 검증을 수행
- 결과 저장: 검증 결과는 반드시 뒤따라오는 BindingResult 객체에 저장
- 오류 처리: bindingResult.hasErrors()를 통해 오류 여부를 확인하고, 오류가 있다면 다시 폼 페이지로 이동

Thymeleaf를 활용해 오류 메시지 표시 (실습 관)

- th:object: 폼 전체에서 사용할 객체 (Model에서 전달된 DTO)를 지정합니다.
- th:field: HTML <input> 요소를 객체의 필드와 자동 바인딩합니다.
- th:errors / #fields.hasErrors(): BindingResult에 담긴 오류 메시지를 자동으로 화면에 출력하여 사용자에게 피드백을 제공합니다.
- 전체 흐름: 사용자 입력 전송 → Spring MVC가 @Valid로 검증 → 오류 발생 시 BindingResult에 저장
  → Thymeleaf 뷰가 th:errors로 오류 메시지 출력.