ReadMe

<엔티티 설계>

회원 – 질문, 답변 : 단방향 연관관계

질문 – 답변 : 양방향 연관관계

@Column(columnDefinition = "TEXT")

DB에서는 제한된 문자열이 기본 타입이다. (varchar(255))

아주 긴 문자열 사용 : columnDefinition = "TEXT"

Application.properties

spring.jpa.hibernate.ddl-auto=create

create : 시작 시 생성, 종료 시 유지

개발 단계에서 사용 // 종료 후 재실행 시 기존 데이터가 없어진다.

update : 기존 테이블을 유지하면서 스키마를 업데이트

개발 단계에서 사용 // 종료 후 재실행 시 기존 데이터가 유지

validate : 테이블 존재 여부만 확인 // 운영에서 사용

create-drop은 애플리케이션 실행 시 테이블 생성, 애플리케이션 종료 시 테이블 삭제 // 운영에서 사용

Redirect

리다이렉트는 단순히 사용자에게 '알리는' 것을 넘어,

**웹 애플리케이션의 안정성(PRG 패턴), 사용자 경험(URL 변경), 기능 확장성(외부 연동), 검색 엔진 최적화(SEO)를 위해 필수적으로 사용한다.**

요청 횟수는 클라이언트 -> 서버, 서버 -> 클라이언트, 클라이언트 -> 서버 총 2회이다.

새로운 요청이므로 요청 객체 공유가 불가능, 주소창 URL이 변경된다.

Redirect를 사용하는 목적

* Post – Redirect – Get (PRG) 패턴 구현
* 클라이언트 브라우저 주소창의 URL 변경 : 현재 상태 반영, 북마크/공유, 뒤로가기 버튼

뒤로가기 버튼은 forward로 구현 시 브라우저 기록에 남지 않아 제대로 동작하지 않는다.

Redirect로 구현 시 새로운 요청이 되기 때문에 브라우저 기록에 남아 버튼이 정상 동작

* 같은 WAS 내 다른 리소스로만 이동 가능한 forward와 달리 다른 WAS의 url로도 이동 가능
* 검색 엔진 최적화 : 오래된 URL을 새 URL로 영구적으로 이동시킬 때 사용

Forward

주소창 URL 변경 x, 요청 횟수는 클라이언트 -> 서버 1회

서버 내부 이동이므로 요청 객체가 공유될 수 있다.

요청 객체 공유 필요 시, 또는 서버 내부에서 다른 리소스로 제어권을 위임하기 위해 사용

회원 탈퇴 검증

@GetMapping(“deleteAccount”)

public String deleteAccount(HttpSession httpsession){

Member loginMember = (Member) httpsession.getAttribute("loginMember");

If(loginMember == null) {

Return “redirect:/signIn”;

}

return “sign/deleteAccountForm”;

}

httpsession.getAttribute("loginMember") 의 리턴타입은 Object 이다 => 다운캐스팅 필요

회원 가입, 로그인, 회원 탈퇴는 Member 엔티티를 직접 사용하기 보다 전용 DTO 객체를 만들어 사용하는 것이 안정적이다.

REST API 설계 원칙에 따르면 리소스 삭제는 @DeleteMapping을 사용하는 것이 맞다.

현실적인 고려 사항 : 모든 브라우저가 Delete 요청을 완전히 지원하지는 않는다.

* 스프링의 지원 : HiddenHttpMethodFilter을 통해 Delete 메서드를 사용하며 문제를 보완할 수 있다.
* 처음에 회원 삭제 매핑을 @DeleteMapping으로 했다가 이후 @PostMapping으로 변경
* 실무에서는 RestApi를 따라서 @DeleteMapping을 쓰기 보다, 호환성이 좋은 @PostMapping을 사용하는 경우가 더 빈번하다.
* Rest 원칙을 따르고 싶을 경우 HiddenHttpMethodFilter를 사용하기
* Application.properties에 spring.mvc.hiddenmethod.filter.enabled=true 추가

@DeleteMapping("/deleteAccount")

public String deleteAccount(@Valid MemberDeleteDto memberDeleteDto, BindingResult bindingResult, HttpSession httpSession) {

*//* 삭제로직

}