설계 및 결과

2024.09.01~2024.09.05

서론

본 개인 프로젝트에서는 spring을 사용하여 tstory와 유사한 기능을 가진 웹 crud 시스템을 개발한다.

spring core, spring mvc, H2, jpa 등의 기술 스택을 사용한다.

요구사항 정리

기능 목록

본 시스템에서 요구되는 기능 목록은 아래와 같다.

1. 회원 관리

   회원 로그인
   회원 가입
   로그아웃

   로그인 상태가 아닌 접근은 /login으로 리다이렉트하도록 처리해야 한다.

2. 회원 별 글 게시판 관리

   글 목록 조회
   타 회원 게시판 조회

3. 글 관리

   글 등록
   글 조회
   글 수정
   글 삭제

4. 댓글 관리

   댓글 작성

5. 회원 별 권한 관리

글 등록, 수정, 삭제는 자신의 게시판과 글에 대해서만 수행이 가능해야 함.

댓글 작성은 모든 게시판에 대해 수행이 가능해야 함.

화면 구성

- 로그인 화면

  name/pw 입력 폼, 제출 버튼
  회원가입 버튼

  잘못된 입력인 경우 메시지 출력

- 회원가입 화면

  name/pw/board name 입력 폼, 제출 버튼
  취소 버튼

  잘못된 입력인 경우 메시지 출력

- 개인 게시판 화면

  게시판 이름, 로그아웃 버튼, 전체 게시판 둘러보기 버튼 (header)
  글 목록, 글 제목 누르면 글 내용 화면으로 이동.
  글쓰기 버튼

- 글 작성 화면

  게시판 이름, 로그아웃 버튼 (header)
  글 제목, 글 내용 입력 폼, 제출 버튼
  취소 버튼

- 글 내용 화면

  게시판 이름, 로그아웃 버튼, 전체 게시판 둘러보기 버튼 (header)
  글 제목, 글 내용
  글 삭제, 수정 버튼
  댓글 제목, 댓글 내용
  댓글 제목, 댓글 내용 작성 폼, 제출 버튼

- 글 수정 화면

  게시판 이름, 로그아웃 버튼, 전체 게시판 둘러보기 버튼 (header)
  글 제목, 글 내용 수정 폼, 제출 버튼
  취소 버튼

- 전체 게시판 목록 화면

전체 게시판 목록, 게시판 이름 누르면 게시판으로 이동.
뒤로 버튼(원래의 게시판으로 이동).

URL 구성

/login
GET : 로그인 화면으로 응답.
POST : HTML 폼으로 전송한 로그인 정보 검사, 로그인하고 /user/{username}으로 리다이렉트.

/signup
GET : 회원가입 화면으로 응답.
POST : HTML 폼으로 전송한 회원가입 정보로 회원가입하고 /login으로 리다이렉트.

/logout
GET : 로그아웃 수행, /login으로 리다이렉트.

/user/{username}
GET : username에 해당하는 개인 게시판 화면으로 응답.

/user/{username}/write
GET : 글 작성 화면으로 응답.
POST : HTML 폼으로 전송한 글 정보로 글 작성하고 /user/{username}으로 리다이렉트.

/user/{username}/{글id}
GET : 글 내용 화면으로 응답.
POST : HTML 폼으로 전송한 댓글 정보로 댓글 작성하고 /user/{username}/{글id}으로 리다이렉트(재출력).

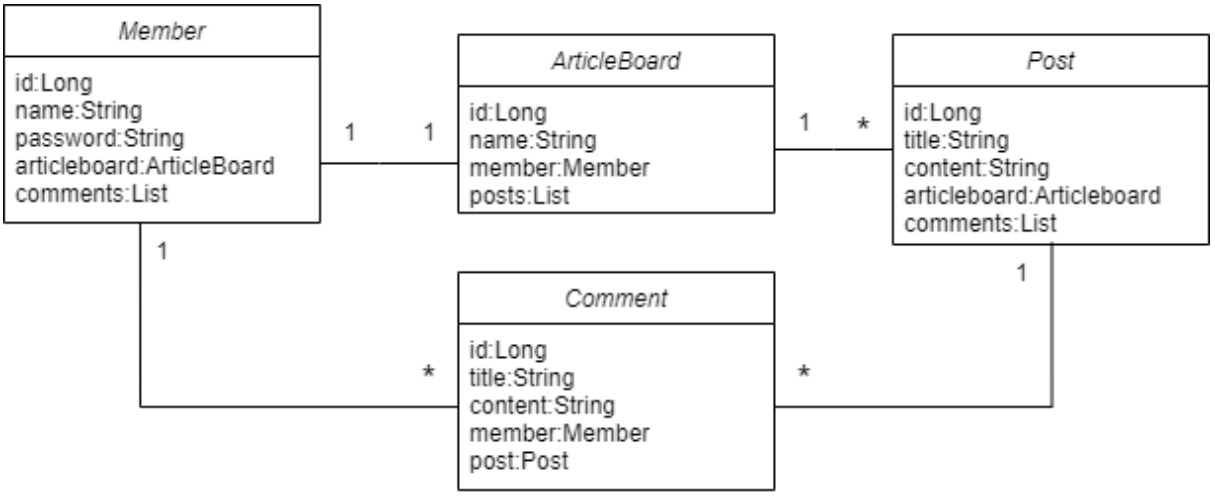
/user/{username}/{글id}/delete
GET : 글 삭제, /user/{username}으로 리다이렉트.

/user/{username}/{글id}/update
GET : 글 수정 화면으로 응답.
POST : HTML 폼으로 전송한 글 정보로 글 수정하고 /user/{username}/{글id}으로 리다이렉트.

/user/all
GET : 전체 게시판 목록 화면으로 응답.

엔티티 모델

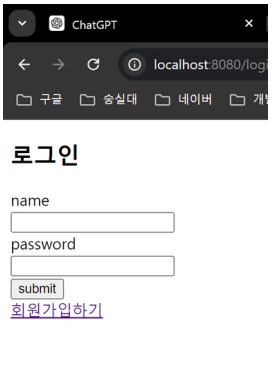
엔티티 모델은 아래와 같음.



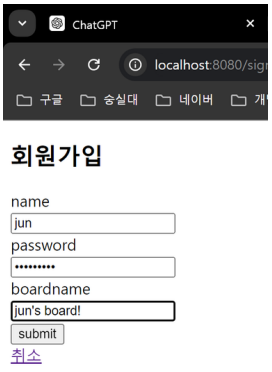
구현 결과

설계 시 작성한 모든 기능의 구현에 성공하였고, 구현 결과 화면은 아래와 같다.

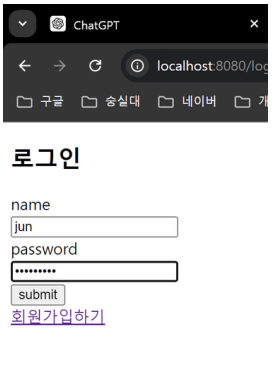
Spring 연습용 프로젝트이므로 웹페이지에 스타일은 따로 적용하지 않았다.



로그인 화면.

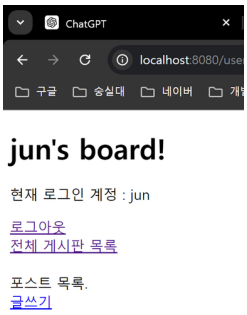


회원가입 화면.

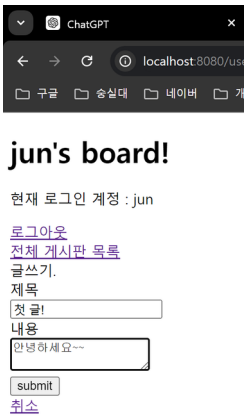


로그인 폼 작성.

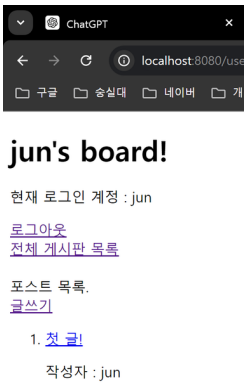
로그인 상태가 아닐 경우 모든 url 접근은 해당 화면으로 리다이렉트됨.



폼 제출 및 회원가입 성공 시 로그인 화면으로 리다이렉트됨.



폼 제출 및 로그인 성공 시 사용자 게시판 화면으로 리다이렉트됨.



사용자 게시판 화면.

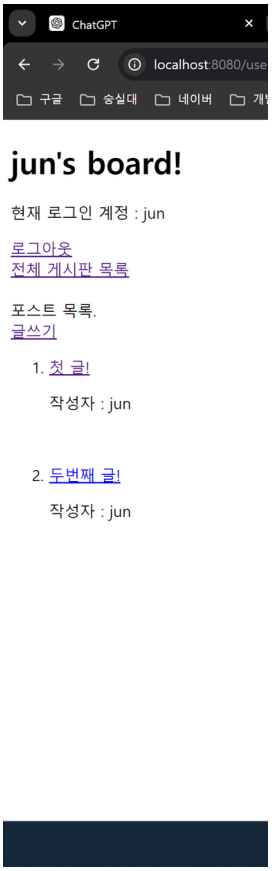
해당 게시판이 현재 로그인 중인 사용자의 게시판일 경우 글쓰기 가능.

로그아웃, 전체 게시판 목록 조회 가능(html 헤더 부분).

게시판에 글쓰기.

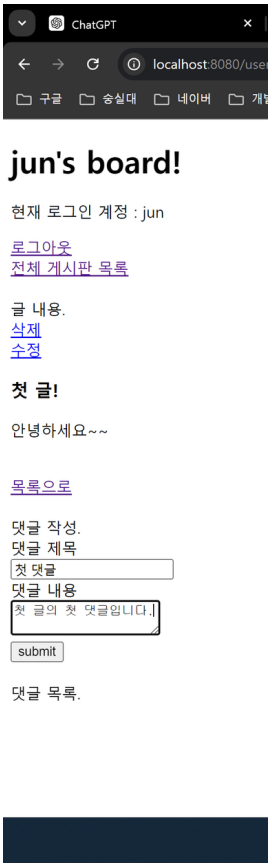
폼 제출 시 사용자 게시판 화면으로 리다이렉트됨.

글쓰기 성공.



여러 개의 글 등록 가능.

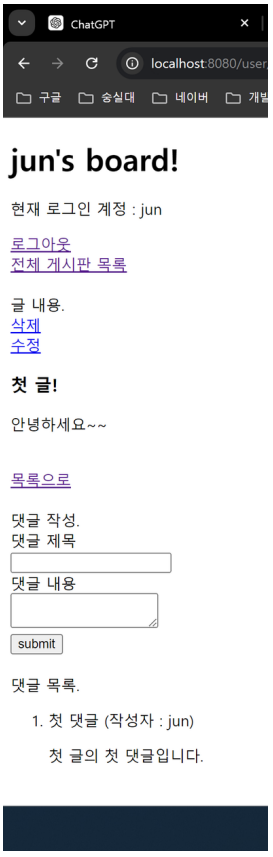
글 제목을 눌러 글 내용 화면으로 이동 가능.



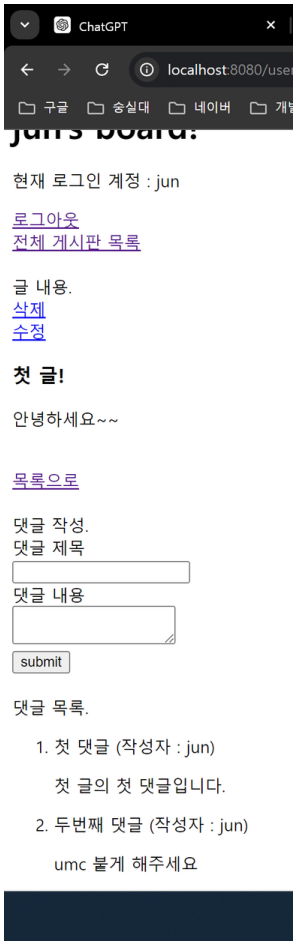
글 내용 화면.

댓글 작성 가능.

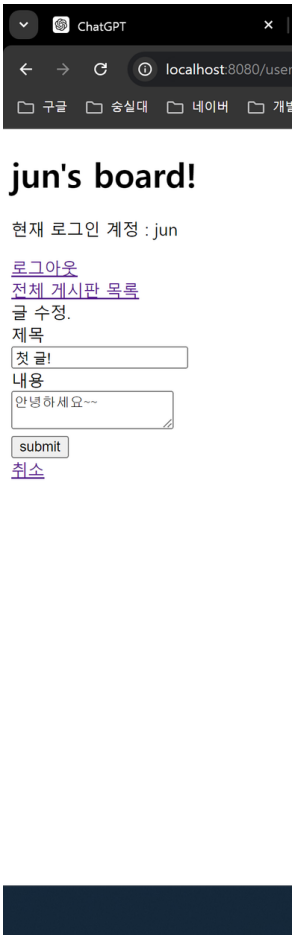
해당 게시판이 현재 로그인 중인 사용자의 게시판일 경우 글 삭제, 수정 가능.



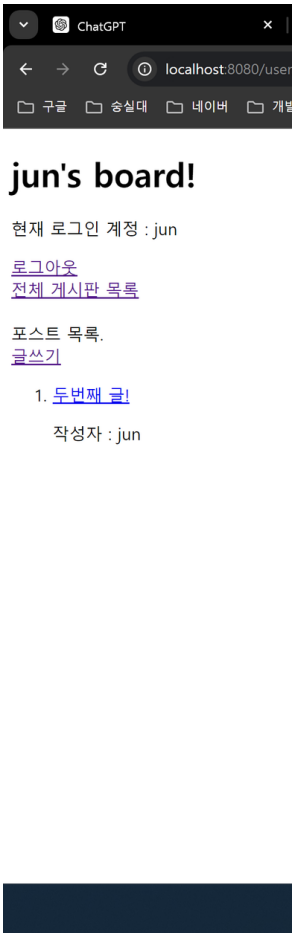
댓글 작성 성공.



여러 개의 댓글 작성 가능.



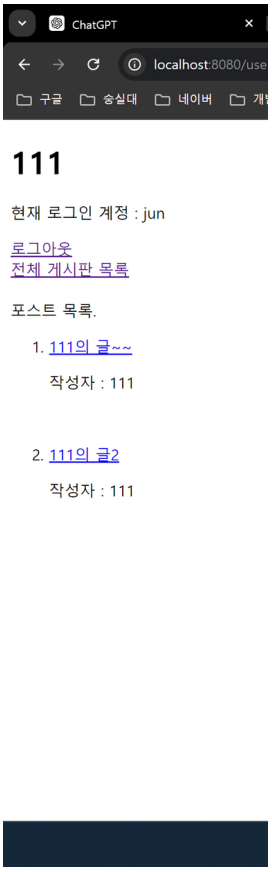
글 수정 화면. 글 수정 가능.



글 삭제 성공.

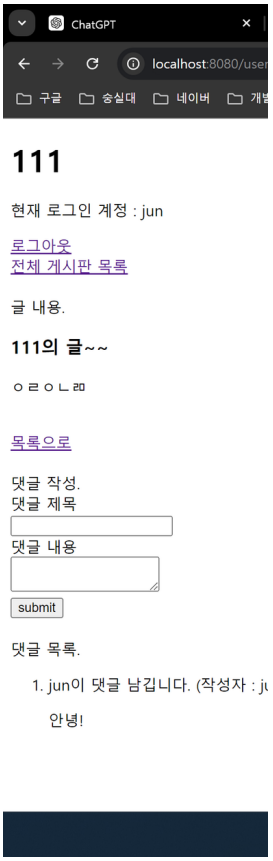
전체 게시판 목록 화면.

다른 사용자의 게시판 접근 가능.



다른 사용자의 게시판 접근 성공.

현재 로그인 중인 사용자의 게시판이 아니므로 글 쓰기 불가.



다른 사용자 게시판에 댓글 작성 가능.

현재 로그인 중인 사용자의 게시판이 아니므로 글 수정, 삭제 불가.

구현 관련 메모

jpa 사용법.

0. db 사용을 위한 환경설정을 한다.
1. JpaRepository<User, Long> 인터페이스를 상속받는 리포지토리 인터페이스 작성한다.
   서비스 계층에서 해당 리포지토리 타입의 빈을 주입받도록 코드를 작성하면 구현체를 자동으로 만들어 주입해준다.
2. 주입받은 빈에 대해 적절한 메소드를 사용한다.

JpaRepository를 사용하는 경우 @Transactional을 지정해줘야 db에 반영된다.

엔티티 처리.

엔티티 처리 시 영속성 전이로 연관된 엔티티를 함께 처리할 수 있다.

https://velog.io/@chiyongs/JPA-영속성-전이-CASCADE

엔티티 객체를 삭제할 때는 연관관계에 있는 객체도 수정해줘야 제대로 삭제된다.
또한 orphanRemoval=true를 지정해야 db에서도 잘 삭제된다.

h2 연결법.

1. application.properties 설정을 한다.
   특히 url 경로를 주의해야 하는 듯.
   처음에는 'jdbc:h2:~/프로젝트명'으로 지정하고, ~/프로젝트명.mv.db 파일이 생성되었는지 확인한다. 이후 'jdbc:h2:tcp://localhost/~/프로젝트명'으로 지정한다.

쿠키 사용법

회원가입 시에 해당 내용을 저장해 둔다.

로그인 성공 시에는 쿠키를 생성하여 응답으로(response) 전송한다.
로그아웃 시에는 쿠키에 대한 종료 시간을 0으로 지정해 응답으로 전송한다.

로그인 시에 쿠키를 검사하여 적용한다.

쿠키에 주요 값을 노출하여 사용한다면 보안상 위험이 존재하므로 세션을 사용한다.

세션 사용법

주요 데이터 대신 세션에 따른 sessionid를 쿠키로 전송한다.
로그인 시에 세션 테이블에 등록하고, sessionid를 쿠키로 전송하는 것.
로그인 시에 sessionid를 전송하면 해당 sessionid에 해당되는 데이터가 사용된다.

서블릿이 제공하는 HttpSession을 사용할 수 있다.

jsessionid는 session의 구분을 위해 사용된다. getSession()은 jsessionid에 해당되는 세션을 찾는다. setAttribute(), getAttribute()는 해당 세션 내에서 데이터를 다룰 때 사용한다.

필터 사용법

필터는 dispatcher servlet의 작업 수행 전에 요청을 처리할 수 있도록 한다.
이에 따라 로그인 여부에 따른 접근 가능 url을 지정할 수 있다.

사용 방법. 1. Filter 인터페이스를 구현한다. 2. WebConfig에서 필터 등록을 한다.