Team11 우연정 4/25

도메인 설계 - 엔티티 클래스 생성 및 연관관계 매핑

- 게시판(Entity: Board)의 속성 및 관계를 정의
- 게시글(Entity: Post)의 속성 및 관계를 정의
-

게시판 기능 구현 요구 사항

- 게시판 조회, 생성, 수정, 삭제 기능 구현
- JdbcTemplate 을 사용하여 DB 에 접근
- 엔티티 클래스 객체 생성 시 Builder 패턴 적용

Board.java

```
package com.elice.boardproject.board.entity;
import com.elice.boardproject.post.entity.Post;

import jakarta.persistence.*;
import lombok.*;

import java.util.ArrayList;
import java.util.List;

@Entity
@Getter
@Setter
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Board {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "board_id")
    private Long id;

    private String title;

    @OneToMany(mappedBy = "board", cascade = CascadeType.ALL, fetch =
FetchType.LAZY)
    private List<Post> posts = new ArrayList<>();


    public void addPost(Post post) {
        posts.add(post);
        post.setBoard(this);
    }
```

```
    public void updateBoard(String title) {
        this.title = title;
    }

}
```

BoardResponseDto.java

```
package com.elice.boardproject.board.dtos;

import com.elice.boardproject.board.entity.Board;
import lombok.Data;

@Data
public class BoardResponseDto {

    private Long id;
    private String title;

    public BoardResponseDto(Board board) {
        this.id = board.getId();
        this.title = board.getTitle();
    }

}
```

BoardRequestDto.java

```
package com.elice.boardproject.board.dtos;

import lombok.Data;

@Data
public class BoardRequestDto {

    private Long id;
    private String title;

}
```

BoardRepository.java

```java
package com.elice.boardproject.board.repository;

import com.elice.boardproject.board.entity.Board;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import java.sql.PreparedStatement;
import java.util.List;

@Repository
@Transactional
public class BoardRepository {

    private final JdbcTemplate jdbcTemplate;

    @Autowired
    public BoardRepository(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    private final RowMapper<Board> boardMapper() {
        return (resultSet, rowNum) -> {
            Board board = new Board();
            board.setId(resultSet.getLong("board_id"));
            board.setTitle(resultSet.getString("title"));
            return board;
        };
    }

    public Long save(Board board) {
        String sql = "INSERT INTO board (title) VALUES (?)";

        KeyHolder keyHolder = new GeneratedKeyHolder();
        jdbcTemplate.update(con -> {
            PreparedStatement ps = con.prepareStatement(sql, new
String[]{"board_id"});
            ps.setString(1, board.getTitle());
            return ps;
        }, keyHolder);

        Long key = keyHolder.getKey().longValue();
        return key;
    }

    public List<Board> findAll() {
        String sql = "SELECT * FROM board";
        return jdbcTemplate.query(sql, boardMapper());
    }

    public Board findById(Long id) {
        String sql = "SELECT * from board WHERE board_id = (?)";
        return jdbcTemplate.queryForObject(sql, boardMapper(), id);
    }
```

```
    public void update(Board board) {
        String sql = "UPDATE board SET title = ? WHERE board_id = ?";
        jdbcTemplate.update(sql, board.getTitle(), board.getId());
    }

    public void delete(Long id) {
        String sql = "DELETE FROM board WHERE board_id = ?";
        jdbcTemplate.update(sql, id);
    }

}
```

BoardService.java

```
package com.elice.boardproject.board.service;

import com.elice.boardproject.board.entity.Board;
import com.elice.boardproject.board.repository.BoardRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
@Transactional(readOnly = true)
@RequiredArgsConstructor
public class BoardService {

    private final BoardRepository boardRepository;

    // 게시판 생성
    @Transactional
    public Long create(Board board) {
        Long createdBoardID = boardRepository.save(board);
        return createdBoardID;
    }

    // 게시판 전체 조회
    public List<Board> findAll() {
        return boardRepository.findAll();
    }

    // 게시판 개별 조회
    public Board findById(Long boardId) {
        return boardRepository.findById(boardId);
    }

    // 게시판 수정
    @Transactional
    public void update(Board board) {
        boardRepository.update(board);
    }
```

```
    // 게시판 삭제
    @Transactional
    public void delete(Long id) {
        boardRepository.delete(id);
    }

}
```

BoardController.java

```
package com.elice.boardproject.board.controller;

import com.elice.boardproject.board.dtos.*;
import com.elice.boardproject.board.entity.Board;
import com.elice.boardproject.board.service.BoardService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller
@RequestMapping("/board")
@RequiredArgsConstructor
public class BoardController {

    private final BoardService boardService;

    @GetMapping("/createBoard")
    public String showCreateForm() {
        return "board/boards";
    }

    @PostMapping("/create")
    public String createBoard(BoardResponseDto boardResponseDto, Model
model) {

            Board board = Board.builder()
                    .title(boardResponseDto.getTitle())
                    .build();
            boardService.create(board);

            return "redirect:/";
    }

    @GetMapping
    public List<Board> getAllBoard() {
        return boardService.findAll();
    }

    @GetMapping("/{id}")
```

```
    public Board getBoardById(@PathVariable("id") Long id) {
        return boardService.findById(id);
    }

    @PostMapping("/modify")
    public ResponseEntity<Board> updateBoard(@RequestBody BoardRequestDto
boardRequestDto) {
        Board board = boardService.findById(boardRequestDto.getId());
        board.updateBoard(boardRequestDto.getTitle());
        boardService.update(board);

        return new ResponseEntity<>(board, HttpStatus.OK);
    }

    @DeleteMapping("/delete/{id}")
    public ResponseEntity<Boolean> deleteBoard(@PathVariable("id") Long id)
{
        boardService.delete(id);

        return new ResponseEntity<>(true, HttpStatus.OK);
    }

}
```