

Spring Boot

21. Board Project (게시판)

Board Project

➤ Board 기능 정의

- ✓ 로그인/회원가입
 - 이메일, 이름, 비밀번호
- ✓ 게시판 등록
 - 신규 게시판 등록 기능
 - 등록 정보 : Id, 타이틀, 내용, 작성자, 등록일, 수정일
- ✓ 게시판 수정
 - 게시판 수정 기능
 - 수정항목 : 타이틀, 내용
- ✓ 게시판 삭제
 - 게시판 삭제 기능
- ✓ 게시판 리스트
 - 화면 정보 : Id, 타이틀, 내용, 작성자, 등록일
 - 페이징 처리
 - 검색 기능
- ✓ 게시판 상세
 - 게시물 상세 조회
 - 조회 카운트

Board Project

➤ 데이터베이스 설계

- ✓ 관계형 데이터베이스에서는 개체(entity)간의 관계로 데이터를 표현
- ✓ 1:1, 1:N, N:1, M:N으로 존재
- ✓ 관계를 구성하는 핵심은 PK(primary key, 주키)와 FK(foreign key, 외래키)
- ✓ 관계를 어떻게 해석하는지가 설계의 관건

회원 데이터

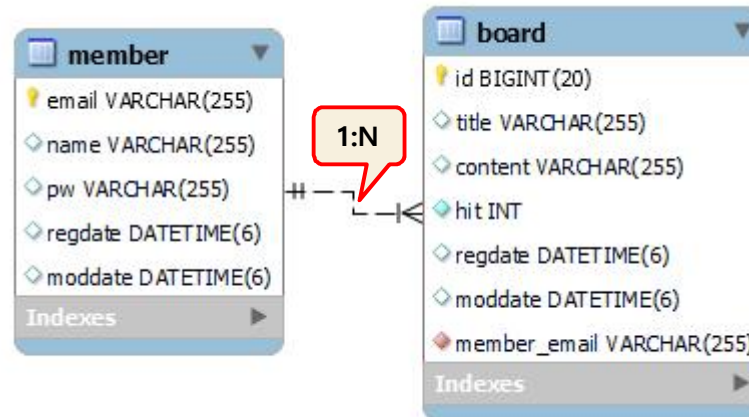
아이디	이름	패스워드
user1	사용자1	111
user2	사용자2	111
user3	사용자3	111
user4	사용자4	111
user5	사용자5	111

게시글 데이터

번호	제목	작성자	내용
1	오늘 날씨가..	user1	
2	집에 오늘 길은..	user2	
3	차가 막혀서..	user3	
4	어이 없는 일이..	user4	
5	감기 조심하세요..	user1	
6	이번 개봉 영화..	user2	
7	오늘 점심에..	user1	

Board Project

➤ ERD(Entity Relationship Diagram)



Board Project

➤ Board API 정의

기능	URL	GET/POST	기능	Redirect URL
로그인	/login	GET	로그인 화면	
	/login	POST	로그인 처리	/list
회원가입	/signup	GET	회원가입 화면	
	/signup	POST	회원가입 처리	/login

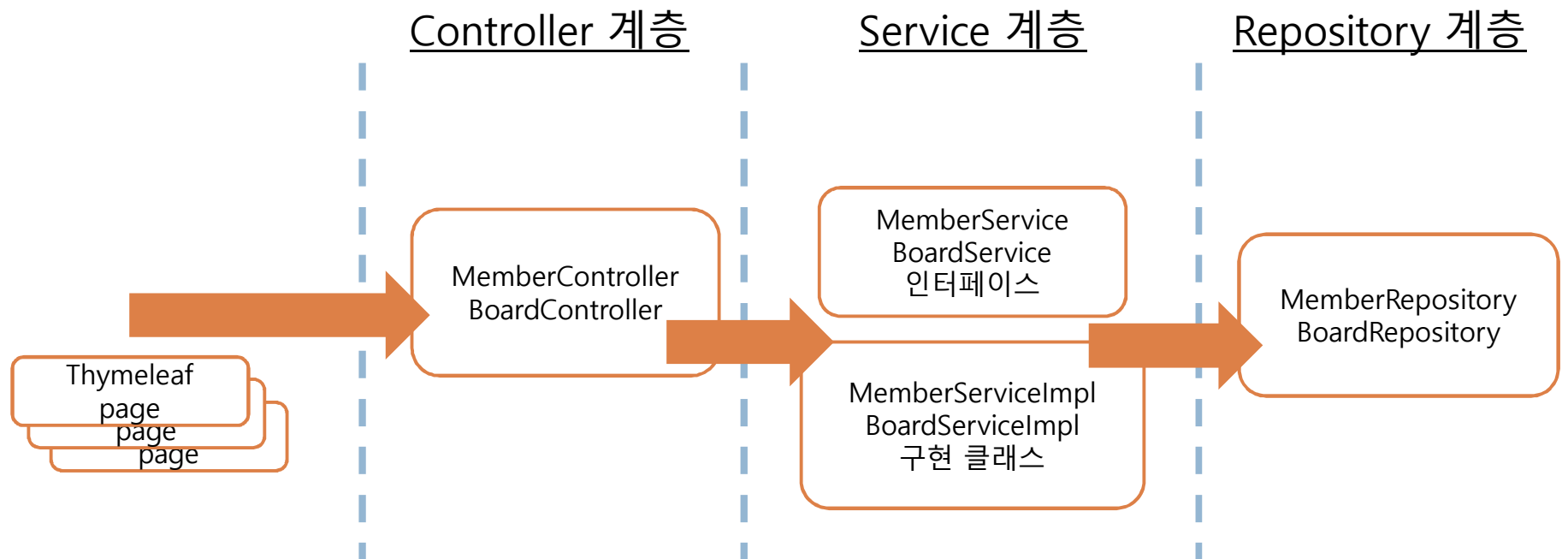
Board Project

➤ Board API 정의

기능	URL	GET/POST	기능	Redirect URL
목록	/list	GET	목록/페이징/검색	
등록	/write	GET	입력 화면	
	/write	POST	등록 처리	/list
조회	/detail	GET	조회 화면	
수정	/modify	GET	수정/삭제 가능 화면	
	/modify	POST	수정 처리	/detail
삭제	/remove	POST	삭제 처리	/list

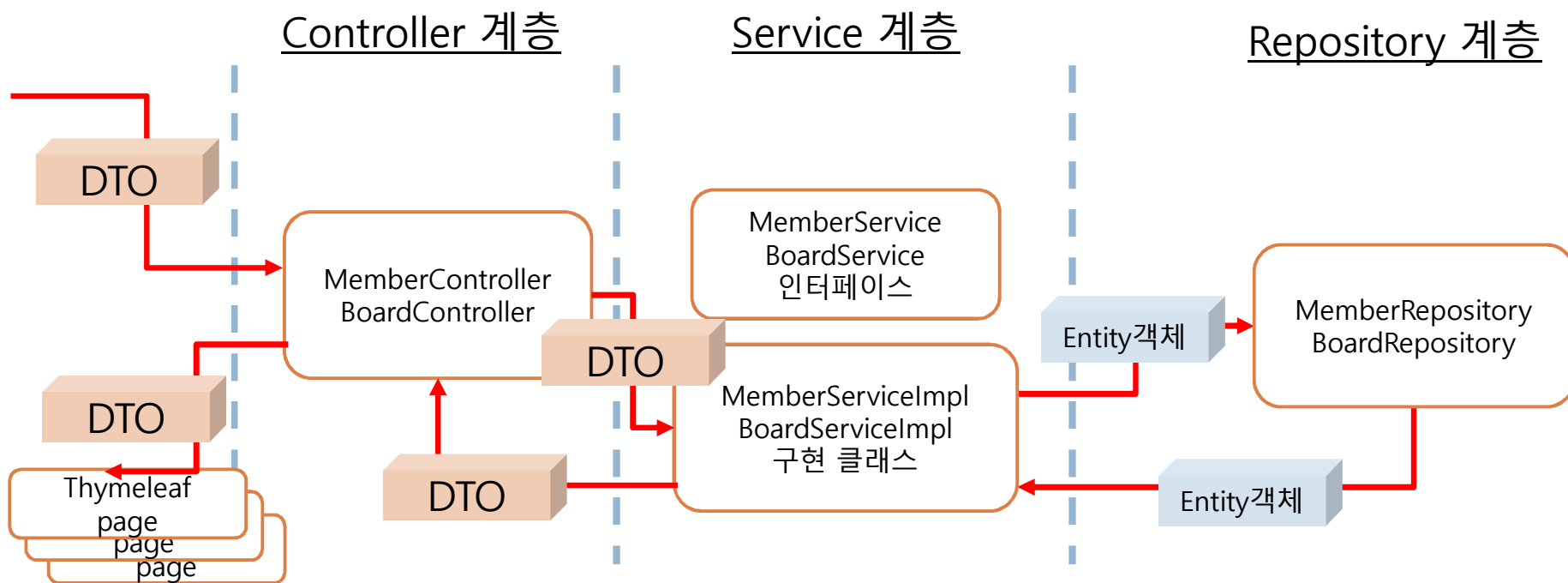
Board Project

➤ 프로젝트 기본 구조



Board Project

➤ 프로젝트 기본 구조



Board Project

➤ 프로젝트 생성

Project
☒ Gradle - Groovy ☐ Gradle - Kotlin
☐ Maven

Language
☒ Java ☐ Kotlin ☐ Groovy

Spring Boot
☐ 3.1.0 (SNAPSHOT) ☐ 3.1.0 (RC2) ☐ 3.1.0 (M2) ☐ 3.0.7 (SNAPSHOT)
☐ 3.0.6 ☐ 2.7.12 (SNAPSHOT) ☒ 2.7.11

Project Metadata
Group
Artifact
Name
Description
Package name
Packaging ☒ Jar ☐ War
Java ☐ 20 ☐ 17 ☒ 11 ☐ 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container. —

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes. —

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate. —

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code. —

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience. —

MySQL Driver SQL
MySQL JDBC driver. —

Board Project

➤ 프로젝트 생성

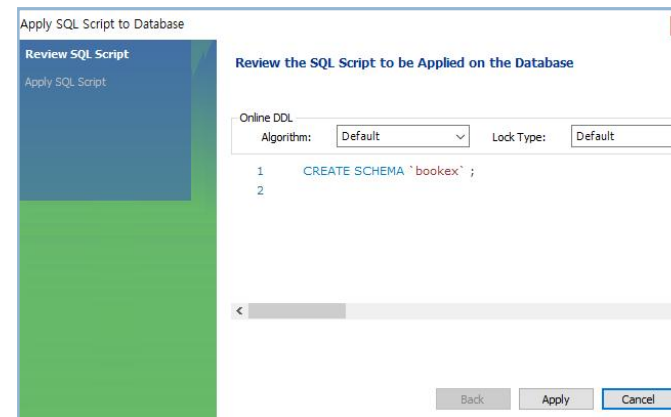
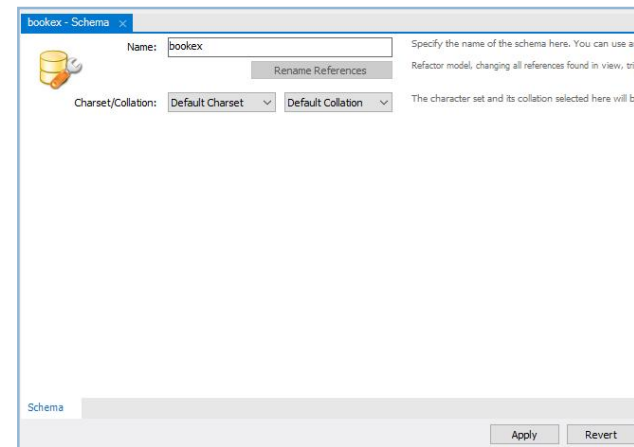
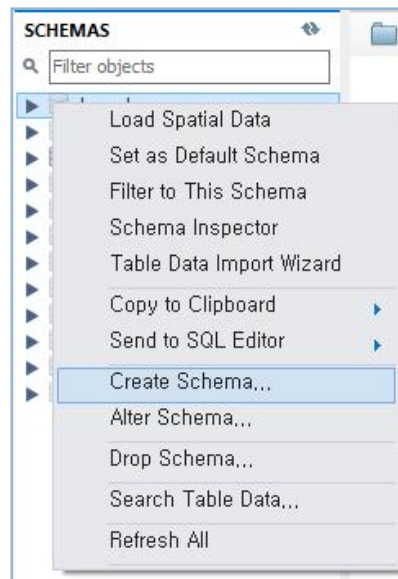
- ✓ build.gradle 라이브러리 추가

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    compileOnly 'org.projectlombok:lombok'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    runtimeOnly 'com.mysql:mysql-connector-j'  
    annotationProcessor 'org.projectlombok:lombok'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
  
    //타임리프에서 dateFormat을 사용하기 위함  
    implementation group: 'org.thymeleaf.extras', name: 'thymeleaf-extras-java8time'  
}
```

Board Project

➤ 프로젝트 생성

- ✓ mysql 스키마 추가 : boardex



Board Project

➤ 프로젝트 생성(application.properties)

```
# mysql 데이터베이스 설정
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/boardex?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=1234

# JPA 설정
spring.jpa.generate-ddl=true
spring.jpa.open-in-view=false
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect

#이미 생성된 결과를 서버에서 캐싱할것인지 설정.
spring.thymeleaf.cache=false

#전체 로그 레벨 설정(기본 info)
logging.level.root=info
#com.example.board 패키지와 그 하위 로그 레벨 설정
logging.level.com.example.board=debug
```

Board Project

➤ 프로젝트 생성

- ✓ 컨트롤러 생성 ('/' API추가)

```
@Slf4j
@Controller
public class BoardController {

    @GetMapping("/")
    public String main(){
        log.info("main");
        return "/list";
    }
}
```

Board Project

➤ 프로젝트 생성

- ✓ Entity(BaseEntity) 생성

```
@MappedSuperclass
@EntityListeners(AuditingEntityListener.class)
@Setter
@Getter
public class BaseEntity {
    @CreatedDate
    @Column(updatable = false)
    private LocalDateTime regDate;

    @LastModifiedDate
    private LocalDateTime modDate;
}
```

```
@EnableJpaAuditing
@SpringBootApplication
public class BoardApplication {

    public static void main(String[] args) {
        SpringApplication.run(BoardApplication.class, args);
    }

}
```

Board Project

➤ 프로젝트 생성

✓ Member Entity 생성

```
@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@ToString(exclude = "boardList ")
public class Member extends BaseEntity{
    @Id
    private String email;
    private String name;
    private String pw;

    @Builder.Default
    @OneToMany(mappedBy = "member", cascade = CascadeType.ALL)
    private List<Board> boardList = new ArrayList<>();
    public void addBoard(Board board){
        boardList.add(board);
        board.setMember(this);
    }
}
```

@ToString 사용 주의
- 연관 관계 필드는
exclude 로 제외
(Exception 발생)

양방향 mappedBy 옵션

- 양방향 매핑 시 필요
- 테이블을 관리할 수 있도록 정하는 옵션
- MappedBy가 정의되지 않은 객체가 주인(Owner)
- 외래키를 가진 객체를 주인으로 정의 (Board)
- Owner가 아닌 객체(Member)는 읽기만 가능

Board Project

➤ 프로젝트 생성

✓ Member Entity 생성

```
@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@ToString(exclude = "boardList")
public class Member extends BaseEntity{
    @Id
    private String email;
    private String name;
    private String pw;

    @Builder.Default
    @OneToMany(mappedBy = "member", cascade = CascadeType.ALL)
    private List<Board> boardList = new ArrayList<>();
    public void addBoard(Board board){
        boardList.add(board);
        board.setMember(this);
    }
}
```

영속성 전이 옵션 (cascade)

- 특정 Entity를 영속 상태로 만들 때 연관된 Entity도 함께 영속상태로 만들고 싶을 때 사용
- 연관 관계 매핑 또는 상속과는 아무 상관없는 독립적인 기능
- 사용 예 : Parent Entity 중심으로 관리하고자 할 때, 즉, Parent Entity를 저장(삭제)할 때 Child Entity도 함께 저장(삭제) 하고자 할 때 사용
- Parent Entity와 Child Entity의 LifeCycle이 동일할 때 사용
- 다른 Entity가 Child Entity와 연관이 있으면 사용하면 안됨

[CASCADE 옵션]

- ALL : 모든 옵션 적용
- PERSIST : 엔티티 영속화 시 연관된 엔티티도 함께 영속화
- REMOVE : 엔티티 삭제 시 연관된 엔티티도 함께 삭제

Board Project

➤ 프로젝트 생성

✓ Board Entity 생성

```
@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@ToString(exclude = "member")
public class Board extends BaseEntity{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    @Column(length = 1500)
    private String content;
    @Builder.Default
    private int hit = 0;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="member_email")
    private Member member;
}
```

@ToString 사용 주의
- 연관 관계 필드는
exclude 로 제외
(Exception 발생)

fetch 속성
FetchType.EAGER (즉시로딩, 디폴트)
FetchType.LAZY (지연로딩)

외래키 참조 필드명 지정

- mappedBy 이름과 동일한 필드명으로 선언해야 함
@OneToMany(mappedBy = "member")

Board Project

➤ 프로젝트 생성

- ✓ Repository 생성

```
public interface MemberRepository extends JpaRepository<Member, String> {  
}
```

```
public interface BoardRepository extends JpaRepository<Board, Long> {  
}
```

Board Project

➤ 프로젝트 생성(QueryDSL)

✓ QueryDSL 라이브러리 추가

- 다양한 상황에 맞게 동적으로 JPQL을 생성하는 방법
- 기본적으로 QueryDSL은 프로젝트 내의 @Entity 어노테이션을 선언한 클래스를 탐색하고, JPAAnnotationProcessor를 사용해 Q 클래스를 생성
- 동적으로 쿼리를 작성할 때 Q도메인을 이용

✓ QueryDSL 장점

- 문자가 아닌 코드로 쿼리를 작성함으로써, 컴파일 시점에 문법 오류를 쉽게 확인
- 자동 완성 등 IDE의 도움을 받을 수 있음
- 동적인 쿼리 작성이 편리
- 쿼리 작성 시 제약 조건 등을 메소드 추출을 통해 재사용할 수 있음

Board Project

➤ 프로젝트 생성(QueryDSL)

```
buildscript {
    ext {
        queryDslVersion = "5.0.0"
    }
}

plugins {
    ... 생략 ...
    //querydsl 추가
    id "com.ewerk.gradle.plugins.querydsl" version "1.0.10"
}
... 생략 ...
dependencies {
    ... 생략 ...
    //querydsl 추가
    implementation "com.querydsl:querydsl-jpa:${queryDslVersion}" // querydsl 라이브러리
    annotationProcessor "com.querydsl:querydsl-apt:${queryDslVersion}" // Querydsl 관련 코드 생성 기능 제공
}
```

Board Project

➤ 프로젝트 생성(QueryDSL) build.gradle

```
... 생략 ...
tasks.named('test'){
    useJUnitPlatform()
}

//querydsl 추가 시작 (위에 plugin 추가 부분과 맞물림)
def querydslDir = "$buildDir/generated/querydsl"
querydsl {
    jpa = true
    querydslSourcesDir = querydslDir
}

sourceSets { // IDE의 소스 폴더에 자동으로 넣어준다.
    main.java.srcDir querydslDir
}

configurations {
    querydsl.extendsFrom compileClasspath // 컴파일이 될 때 같이 수행
}

compileQuerydsl {
    options.annotationProcessorPath = configurations.querydsl // Q파일을 생성해준다.
}
//querydsl 추가 끝
```

Board Project

➤ 프로젝트 생성(QueryDSL)

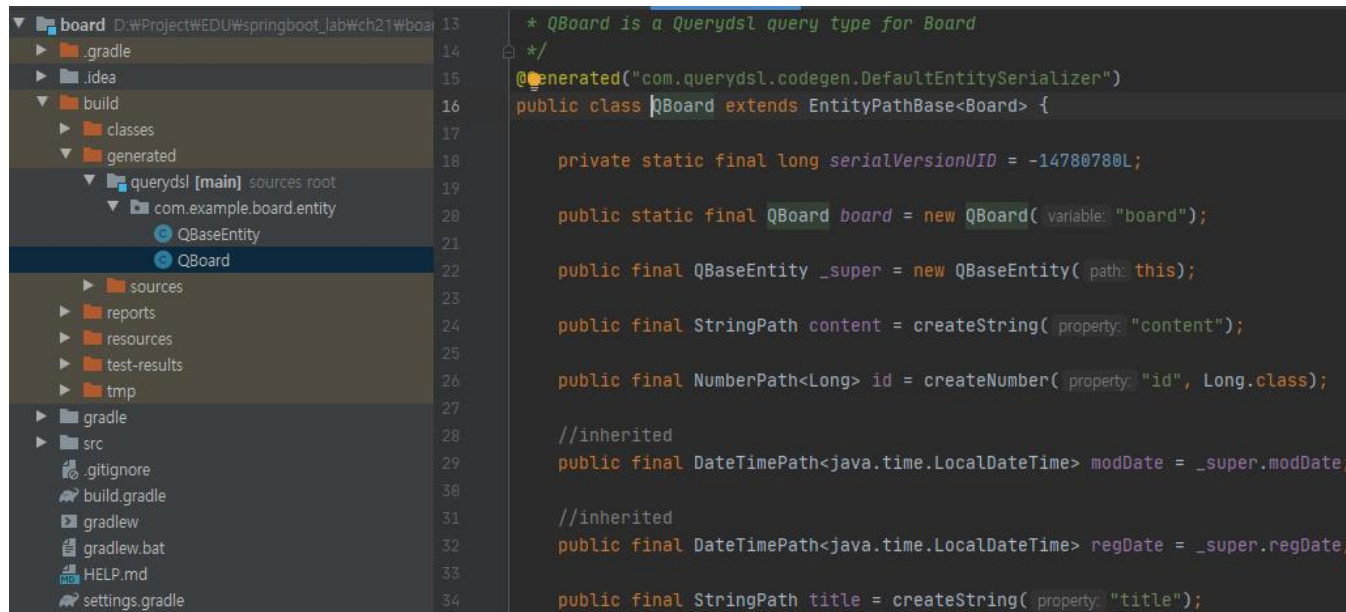
- ✓ QueryDSL 빌드

```
▶ compileQuerydsl {  
    options.annotationProcessorPath = configurations.querydsl // Q파일을 생성해준다.  
}  
//querydsl 추가 끝
```

Board Project

➤ 프로젝트 생성(QueryDSL)

- ✓ QueryDSL Q도메인 클래스 자동 생성됨. Board Entity가 클래스로 생성
- ✓ 개발자가 코드 수정 불가, QBoard 를 이용 동적 쿼리 작성 가능



The screenshot shows an IDE with a project structure on the left and the source code of the generated `QBoard` class on the right. The project structure includes a `board` directory with subdirectories like `.gradle`, `.idea`, `build`, `classes`, `generated`, `querydsl [main] sources root`, `com.example.board.entity`, `QBaseEntity`, `QBoard`, `sources`, `reports`, `resources`, `test-results`, `tmp`, `gradle`, `src`, `.gitignore`, `build.gradle`, `gradlew`, `gradlew.bat`, `HELP.md`, and `settings.gradle`. The `QBoard` class is highlighted in the `generated` directory. The source code of `QBoard` is shown on the right, starting with a comment: `* QBoard is a Querydsl query type for Board`. The class is annotated with `@Generated("com.querydsl.codegen.DefaultEntitySerializer")` and extends `EntityPathBase<Board>`. It contains several static final fields: `serialVersionUID`, `board`, `_super`, `content`, `id`, `modDate`, `regDate`, and `title`. The `board` field is initialized with `new QBoard(variable: "board")`. The `_super` field is initialized with `new QBaseEntity(path: this)`. The `content` field is initialized with `createString(property: "content")`. The `id` field is initialized with `createNumber(property: "id", Long.class)`. The `modDate` and `regDate` fields are initialized with `_super.modDate` and `_super.regDate` respectively. The `title` field is initialized with `createString(property: "title")`.

```
13 * QBoard is a Querydsl query type for Board
14 */
15 @Generated("com.querydsl.codegen.DefaultEntitySerializer")
16 public class QBoard extends EntityPathBase<Board> {
17
18     private static final long serialVersionUID = -14780780L;
19
20     public static final QBoard board = new QBoard(variable: "board");
21
22     public final QBaseEntity _super = new QBaseEntity(path: this);
23
24     public final StringPath content = createString(property: "content");
25
26     public final NumberPath<Long> id = createNumber(property: "id", Long.class);
27
28     //inherited
29     public final DateTimePath<java.time.LocalDateTime> modDate = _super.modDate;
30
31     //inherited
32     public final DateTimePath<java.time.LocalDateTime> regDate = _super.regDate;
33
34     public final StringPath title = createString(property: "title");
```


Board Project

➤ 프로젝트 생성(QueryDSL)

- ✓ QuerydslPredicatedExecutor 이용 QueryDSL 구현

```
public interface BoardRepository extends JpaRepository<Board, Long>,  
    QuerydslPredicateExecutor<Board>{  
}
```

Board Project

➤ 프로젝트 생성(QueryDSL)

- ✓ QuerydslRepositorySupport 이용 QueryDSL 구현(좀더 복잡한 쿼리 작성 가능)

```
public interface BoardCRepository {  
  
}
```

```
public class BoardCRepositoryImpl extends QuerydslRepositorySupport implements BoardCRepository {  
  
    public BoardCRepositoryImpl() {super(Board.class);}  
  
}
```

```
public interface BoardRepository extends JpaRepository<Board, Long>, BoardCRepository {  
  
}
```

Board Project

➤ 엔티티 Insert 테스트

```
@SpringBootTest
public class MemberRepositoryTest {

    @Autowired MemberRepository memberRepository;

    @Test
    public void insertMember(){
        IntStream.rangeClosed(1, 10).forEach(i -> {
            Member member = Member.builder().email("user"+i+"@gmail.com")
                .name("user"+i)
                .pw("user"+i)
                .build();
            memberRepository.save(member);
        });
    }
}
```

email	mod_date	reg_date	name	pw
user9@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user9	user9
user8@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user8	user8
user7@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user7	user7
user6@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user6	user6
user5@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user5	user5
user4@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user4	user4
user3@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user3	user3
user2@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user2	user2
user1@gmail.com	2023-05-11 12:42:59	2023-05-11 12:42:59	user1	user1
user10@gmail.com	2023-05-11 12:43:00	2023-05-11 12:43:00	user10	user10

Board Project

➤ 엔티티 Insert 테스트

```
@SpringBootTest
public class BoardRepositoryTest {
    @Autowired MemberRepository memberRepository;
    @Autowired BoardRepository boardRepository;

    @Test
    public void insertBoard(){
        IntStream.rangeClosed(1, 100).forEach(i -> {
            int num = (int) Math.ceil(Math.random()*10);
            String email = "user"+num+"@gmail.com";
            Optional<Member> optionalMemeber = memberRepository.findByEmail(email);
            if(optionalMemeber.isPresent()){
                Board entity = Board.builder()
                    .title("Title"+i)
                    .content("Content" + i)
                    .member(optionalMemeber.get())
                    .build();
                boardRepository.save(entity);
            }
        });
    }
}
```

id	mod_date	reg_date	content	title	member_email
1	2023-05-11 13:01:28	2023-05-11 13:01:28	Content 1	Title1	user5@gmail.com
2	2023-05-11 13:01:28	2023-05-11 13:01:28	Content 2	Title2	user5@gmail.com
3	2023-05-11 13:01:28	2023-05-11 13:01:28	Content 3	Title3	user4@gmail.com
4	2023-05-11 13:01:29	2023-05-11 13:01:29	Content 4	Title4	user5@gmail.com
5	2023-05-11 13:01:29	2023-05-11 13:01:29	Content 5	Title5	user9@gmail.com
6	2023-05-11 13:01:29	2023-05-11 13:01:29	Content 6	Title6	user5@gmail.com
7	2023-05-11 13:01:29	2023-05-11 13:01:29	Content 7	Title7	user3@gmail.com
8	2023-05-11 13:01:29	2023-05-11 13:01:29	Content 8	Title8	user10@gmail.com
9	2023-05-11 13:01:29	2023-05-11 13:01:29	Content 9	Title9	user5@gmail.com
10	2023-05-11 13:01:29	2023-05-11 13:01:29	Content 10	Title10	user4@gmail.com
11	2023-05-11 13:01:29	2023-05-11 13:01:29	Content 11	Title11	user1@gmail.com

Board Project

➤ 엔티티 Update 테스트

```
@Test
public void updateBoard(){
    Optional<Board> optionalBoard = boardRepository.findById(1L);
    if(optionalBoard.isPresent()){
        Board board = optionalBoard.get();
        board.setTitle("change Title");
        board.setContent("change Content");
        boardRepository.save(board);
    }
}
```

id	mod_date	reg_date	content	title	member_email
1	2023-05-11 13:06:42	2023-05-11 13:01:28	change Content	change Title	user5@gmail.com

Board Project

➤ Querydsl 테스트

- ✓ title 내에 '1'이라는 글자가 포함된 데이터 검색

```
@Test
public void queryDSLTest(){
    Pageable pageable = PageRequest.of(0, 10, Sort.by("id").descending());
    QBoard qBoard = QBoard.board;
    String keyword = "1";
    BooleanBuilder builder = new BooleanBuilder();
    BooleanExpression expression = qBoard.title.contains(keyword);
    builder.and(expression);
    Page<Board> result = boardRepository.findAll(builder, pageable);
    result.stream().forEach(board -> {
        System.out.println(board);
    });
}
```

Q도메인 클래스 참조

BooleanBuilder 에
BooleanExpression
조건문 포함

BooleanBuilder 생성

BooleanExpression
조건문 생성

검색

```
Board(id=100, title=Title100, content=Content 100, hit=0)
Board(id=91, title=Title91, content=Content 91, hit=0)
Board(id=81, title=Title81, content=Content 81, hit=0)
Board(id=71, title=Title71, content=Content 71, hit=0)
Board(id=61, title=Title61, content=Content 61, hit=0)
Board(id=51, title=Title51, content=Content 51, hit=0)
Board(id=41, title=Title41, content=Content 41, hit=0)
Board(id=31, title=Title31, content=Content 31, hit=0)
Board(id=21, title=Title21, content=Content 21, hit=0)
Board(id=19, title=Title19, content=Content 19, hit=0)
```

Board Project

➤ Querydsl 테스트

- ✓ 여러 개의 조건이 결합된 형태 검색(타이틀, 내용, 작성자중 키워드 포함 데이터)

```
@Test
public void queryDSLTest2() {
    //키워드 "1" 이 타이틀, 내용, 작성자에 포함된 데이터 검색
    Pageable pageable = PageRequest.of(0, 10, Sort.by("id").descending());
    QBoard qBoard = QBoard.board;
    String keyword = "1";
    BooleanBuilder builder = new BooleanBuilder();
    BooleanExpression exTitle = qBoard.title.contains(keyword);
    BooleanExpression exContent = qBoard.content.contains(keyword);
    BooleanExpression exWriter = qBoard.member.name.contains(keyword);
    BooleanExpression exAll = exTitle.or(exContent).or(exWriter);
    builder.and(exAll);
    builder.and(qBoard.id.gt(0L));
    Page<Board> result = boardRepository.findAll(builder, pageable);
    result.stream().forEach(board -> {
        System.out.println(board);
    });
}
```

Q도메인 클래스 참조

BooleanBuilder 생성

BooleanBuilder 에
BooleanExpression
조건문 포함

BooleanExpression
조건문 생성

검색

```
Board(id=100, title=Title100, content=Content 100, hit=0)
Board(id=91, title=Title91, content=Content 91, hit=0)
Board(id=81, title=Title81, content=Content 81, hit=0)
Board(id=71, title=Title71, content=Content 71, hit=0)
Board(id=61, title=Title61, content=Content 61, hit=0)
Board(id=51, title=Title51, content=Content 51, hit=0)
Board(id=41, title=Title41, content=Content 41, hit=0)
Board(id=31, title=Title31, content=Content 31, hit=0)
Board(id=21, title=Title21, content=Content 21, hit=0)
Board(id=19, title=Title19, content=Content 19, hit=0)
```

Board Project

➤ MemberDTO(Data Transfer Object) 생성

- ✓ 클라이언트와 데이터 연동은 순수 DTO 객체로 주고 받아야 함
- ✓ JPA 엔티티 객체로 클라이언트와 연동은 보안 상 권장하지 않음

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@ToString
public class MemberDTO {
    private String email;
    private String name;
    private String pw;
    private LocalDateTime regDate;
    private LocalDateTime modDate;
}
```


Board Project

➤ BoardDTO(Data Transfer Object) 생성

- ✓ 클라이언트와 데이터 연동은 순수 DTO 객체로 주고 받아야 함
- ✓ JPA 엔티티 객체로 클라이언트와 연동은 보안 상 권장하지 않음

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@ToString
public class BoardDTO {
    private Long id;
    private String title;
    private String content;
    private LocalDateTime regDate;
    private LocalDateTime modDate;

    private String email;
    private String writer;
}
```

Board Project

➤ MemberService 생성

- ✓ 서비스 인터페이스 생성

```
public interface MemberService {  
    // 등록  
    String register(MemberDTO dto);  
  
    //dto => entity 변환  
    default Member dtoToEntity(MemberDTO dto){  
        Member entity = Member.builder()  
            .email(dto.getEmail())  
            .name(dto.getName())  
            .pw(dto.getPw())  
            .build();  
        return entity;  
    }  
}
```

회원 등록

DTO => Entity 변환

Board Project

➤ MemberService 생성

✓ 서비스 인터페이스 구현

```
@Service
@RequiredArgsConstructor // lombok 의존성 자동주입
public class MemberServiceImpl implements MemberService{

    private final MemberRepository memberRepository;

    @Override public String register(MemberDTO dto) {
        //이메일 중복 예외처리
        if(memberRepository.findById(dto.getEmail()).isPresent()){
            return null;
        }
        //신규 등록
        Member member = dtoToEntity(dto);
        member = memberRepository.save(member);
        return member.getEmail();
    }
}
```

Board Project

➤ MemberService 서비스 테스트

```
@SpringBootTest
public class MemberServiceTest {

    @Autowired MemberService memberService;

    @Test
    @Rollback(value = false)
    public void registerTest(){
        MemberDTO dto = MemberDTO.builder()
            .email("test@gmail.com")
            .name("test")
            .pw("1234")
            .build();
        memberService.register(dto);
    }
}
```

email	mod_date	reg_date	name	pw
test@gmail.com	2023-05-11 14:36:03	2023-05-11 14:36:03	test	1234

Board Project

➤ BoardService 생성

- ✓ 서비스 인터페이스 생성

```
public interface BoardService {  
    // 등록  
    Long register(BoardDTO dto);  
  
    //dto => entity 변환  
    default Board dtoToEntity(BoardDTO dto){  
        Board entity = Board.builder()  
            .id(dto.getId())  
            .title(dto.getTitle())  
            .content(dto.getContent())  
            .hit(dto.getHit())  
            .build();  
        return entity;  
    }  
}
```

목록 등록

DTO => Entity 변환

Board Project

➤ BoardService 생성

✓ 서비스 인터페이스 생성

```
//entity => dto 변환
default BoardDTO entityToDto(Board entity){
    BoardDTO dto = BoardDTO.builder()
        .id(entity.getId())
        .title(entity.getTitle())
        .content(entity.getContent())
        .hit(entity.getHit())
        .email(entity.getMember().getEmail())
        .writer(entity.getMember().getName())
        .regDate(entity.getRegDate())
        .build();
    return dto;
}

//entity리스트 => dto리스트 변환
default List<BoardDTO> toList(List<Board> list){
    return list.stream().map(entity -> entityToDto(entity)).collect(Collectors.toList());
}
```

Entity => DTO 변환

Entity 리스트 => DTO 리스트 변환

Board Project

➤ BoardService 생성

✓ 서비스 인터페이스 구현

```
@Service
@RequiredArgsConstructor // 의존성 자동 주입
public class BoardServiceImpl implements BoardService {

    private final MemberRepository memberRepository;
    private final BoardRepository boardRepository;

    @Override
    public Long register(BoardDTO dto) {
        Long id = null;
        Optional<Member> optionalMember = memberRepository.findById(dto.getEmail());
        if(optionalMember.isPresent()){
            Board entity = dtoToEntity(dto);
            //Member 주입
            Member member = optionalMember.get();
            entity.setMember(member);
            entity = boardRepository.save(entity);
            id = entity.getId();
        }
        return id;
    }
}
```

Member Entity 조회

작성 게시물 DTO => Entity 변환

Member 주입하여 저장

Board Project

➤ BoardService 서비스 테스트

```
@SpringBootTest
public class BoardServiceTest {

    @Autowired BoardService boardService;

    @Test
    @Rollback(value = false)
    public void registerTest(){
        BoardDTO dto = BoardDTO.builder()
            .title("Test Title")
            .content("Test Content")
            .email("test@gmail.com")
            .build();
        boardService.register(dto);
    }
}
```

id	mod_date	reg_date	content	title	member_email	hit
101	2023-05-11 14:59:37	2023-05-11 14:59:37	Test Content	Test Title	test@gmail.com	0

Board Project

- 회원가입 처리(등록화면 구성, 부트스트랩 이용)
 - ✓ form 검색

회원가입

이메일

비밀번호

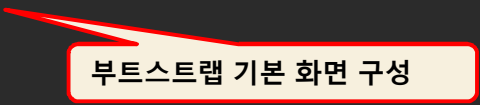
이름

Board Project

➤ 회원가입 처리(등록화면 구성, 부트스트랩 이용)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>회원가입</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css"
rel="stylesheet" >
</head>
<body>
  <div class="container">
    <h1>회원가입</h1>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</body>
</html>
```



부트스트랩 기본 화면 구성

Board Project

➤ 회원가입 처리(등록화면 구성, 부트스트랩 이용)

```
<div class="container">
  <h1>회원가입</h1>
  <form action="/signup" method="post">
    <div class="mb-3 col-3">
      <label for="email" class="form-label">이메일</label>
      <input type="text" class="form-control" id="email" name="email" placeholder="이메일">
    </div>
    <div class="mb-3 col-6">
      <label for="pw" class="form-label">비밀번호</label>
      <input type="password" class="form-control" id="pw" name="pw" placeholder="비밀번호">
    </div>
    <div class="mb-3 col-6">
      <label for="name" class="form-label">이름</label>
      <input type="text" class="form-control" id="name" name="name" placeholder="이름">
    </div>
    <button type="submit" class="btn btn-primary mt-3">회원가입</button>
    <button type="button" class="btn btn-secondary mt-3">취소</button>
  </form>
</div>
```

Board Project

➤ 회원가입 처리

- ✓ 컨트롤러 '/signup' GET API 추가

```
@Slf4j
@Controller
public class MemberController {

    private MemberService memberService;

    @Autowired
    public MemberController(MemberService memberService){
        this.memberService = memberService;
    }

    //신규생성 화면
    @GetMapping("/signup")
    public String signup(){
        log.info("signup");
        return "/signup";
    }
}
```

회원가입 화면 호출

Board Project

➤ 회원가입 처리

- ✓ 컨트롤러 '/ signup' POST API 추가

```
//신규저장
@PostMapping("/signup")
public String register(MemberDTO dto, RedirectAttributes redirectAttributes){
    log.info("register dto:" + dto);
    String email = memberService.register(dto);
    if(email!=null && email.isEmpty()==false){
        redirectAttributes.addFlashAttribute("msg", "회원가입이 되었습니다");
    }
    return "redirect:/login";
}
```

회원 정보 저장

Board Project

- 회원로그인 처리(등록화면 구성, 부트스트랩 이용)
 - ✓ form 검색

로그인

이메일

비밀번호

Board Project

➤ 회원로그인 처리(등록화면 구성, 부트스트랩 이용)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>로그인</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css"
rel="stylesheet" >
</head>
<body>
  <div class="container">
    <h1>로그인</h1>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</body>
</html>
```



부트스트랩 기본 화면 구성

Board Project

➤ 회원로그인 처리(등록화면 구성, 부트스트랩 이용)

```
<div class="container">
  <h1>로그인</h1>
  <form action="/login" method="post">
    <div class="mb-3 col-3">
      <label for="email" class="form-label">이메일</label>
      <input type="text" class="form-control" id="email" name="email" placeholder="이메일">
    </div>
    <div class="mb-3 col-6">
      <label for="pw" class="form-label">비밀번호</label>
      <input type="password" class="form-control" id="pw" name="pw" placeholder="비밀번호">
    </div>
    <button type="submit" class="btn btn-primary mt-3">로그인</button>
    <button type="button" class="btn btn-secondary mt-3" onclick="location.href='/signup'">회원가입</button>
  </form>
</div>
```


Board Project

➤ 회원로그인 처리

- ✓ MemberRepository 로그인 처리 API 추가

```
public interface MemberRepository extends JpaRepository<Member, String> {  
    //Optional<Member> findByEmailAndPw(String email, String pw);  
    @Query("select m from Member m where m.email=:email and m.pw=:pw")  
    Optional<Member> findByEmailAndPw(@Param("email") String email, @Param("pw") String pw);  
}
```

로그인 체크 API

Board Project

➤ 회원로그인 처리

- ✓ MemberService 로그인 처리 API 추가

```
// 로그인  
MemberDTO login(MemberDTO dto);
```

```
@Override public MemberDTO login(MemberDTO dto) {  
    Optional<Member> optionalMember = memberRepository.findByEmailAndPw(dto.getEmail(), dto.getPw());  
    if(optionalMember.isPresent()){  
        return entityToDto(optionalMember.get());  
    }  
    return null;  
}
```

로그인 체크 API

Board Project

➤ 회원로그인 처리

- ✓ 컨트롤러 '/login' GET API 추가

```
@Slf4j
@Controller
public class MemberController {

    private MemberService memberService;

    @Autowired
    public MemberController(MemberService memberService){
        this.memberService = memberService;
    }

    //로그인 화면
    @GetMapping("/login")
    public String login(){
        log.info(" login ");
        return "/ login ";
    }
}
```

로그인 화면 호출

Board Project

➤ 회원로그인 처리

- ✓ 컨트롤러 '/login' POST API 추가

```
//로그인 처리
@PostMapping("/login")
public String doLogin(MemberDTO dto, RedirectAttributes redirectAttributes, HttpServletRequest request){
    log.info("doLogin dto:" + dto);
    MemberDTO memberDTO = memberService.login(dto);
    if(memberDTO==null){
        redirectAttributes.addFlashAttribute("msg", "로그인을 하지 못했습니다");
        return "redirect:/login";
    }

    //세션정보 저장
    HttpSession httpSession = request.getSession();
    httpSession.setAttribute("member", memberDTO);
    return "redirect:/list";
}
```

로그인 체크

로그인 실패 시
로그인 페이지 이동

사용자 정보
HttpSession 정보에 저장

로그인 성공 시
메인 페이지 이동

Board Project

➤ 회원로그인 처리

- ✓ 컨트롤러 '/logout' GET API 추가

```
//로그아웃 처리
@GetMapping("/logout")
public String logout(HttpServletRequest request){
    log.info("logout");

    HttpSession httpSession = request.getSession();
    httpSession.invalidate(); //세션 초기화
    return "redirect:/list";
}
```

사용자 세션 정보 초기화

Board Project

➤ 목록 처리

✓ Get 방식으로 파라미터 전달

- 페이지번호, 검색 조건 전달용 DTO 작성 필요 (PageRequestDTO)
- 페이지 정보 파라미터로 전달 처리

✓ 조회 결과 DTO 리스트로 출력

- Pageable DTO 타입 결과 리스트
- Entity 리스트 => DTO 리스트로 변환 처리 필요

✓ 페이징 처리

- 화면 페이지 처리

Board Project

- 게시판 목록 처리(PageRequestDTO 처리)
 - ✓ PageRequestDTO 작성 (페이지번호, 검색 조건 요청 전달용 DTO)

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class PageRequestDTO {
    @Builder.Default
    private int page = 1;
    @Builder.Default
    private int size = 10;

    public Pageable getPageable(Sort sort){
        return PageRequest.of(this.page-1, this.size, sort);
    }
}
```

Board Project

➤ 게시판 목록 처리(PageResultDTO 처리)

- ✓ PageResultDTO 작성 (조회 결과 리스트 및 페이징 정보 처리)

```
@Data
public class PageResultDTO<DTO, EN> {
    //DTO리스트
    private List<DTO> dtoList;
    //총 페이지 번호
    private int totalPage;
    //현재 페이지 번호
    private int page;
    // 목록 사이즈
    private int size;
    // 시작 페이지 번호, 끝 페이지 번호
    private int start, end;
    // 이전, 다음
    private boolean prev, next;
    // 페이지 번호 목록
    private List<Integer> pageList;
    ... 생략 ...
}
```


Board Project

➤ 게시판 목록 처리(PageResultDTO 처리)

- ✓ PageResultDTO 작성 (조회 결과 리스트 및 페이징 정보 처리)

```
@Data
public class PageResultDTO<DTO, EN> {
    ... 생략 ...
    public PageResultDTO(Page<EN> result, Function<EN, DTO> fn) {
        dtoList = result.stream().map(fn).collect(Collectors.toList());
        totalPage = result.getTotalPages();
        makePageList(result.getPageable());
    }

    private void makePageList(Pageable pageable){
        this.page = pageable.getPageNumber() + 1; // 0부터 시작하므로 1을 추가
        this.size = pageable.getPageSize();
        //temp end page
        int tempEnd = (int)(Math.ceil(page/10.0)) * 10;
        start = tempEnd - 9; // 시작 페이지 번호
        prev = start > 1; //이전
        end = totalPage > tempEnd ? tempEnd: totalPage; // 끝 페이지 번호
        next = totalPage > tempEnd; //다음
        pageList = IntStream.rangeClosed(start, end).boxed().collect(Collectors.toList());
    }
}
```

조회 결과

페이징 정보 추출

현재 페이지를 기준으로 화면에 출력되어야 하는 마지막 페이지 번호를 우선 처리

Board Project

- 게시판 목록 처리(BoardService 처리)
 - ✓ 목록 조회 서비스 API 추가(PageResultDTO 사용)

```
//목록 조회  
PageResultDTO<BoardDTO, Board> getList(PageRequestDTO requestDTO);
```

```
@Override public PageResultDTO<BoardDTO, Board> getList(  
    PageRequestDTO requestDTO) {  
    Pageable pageable = requestDTO.getPageable(Sort.by("id").descending());  
    Page<Board> result = boardRepository.findAll(pageable);  
    Function<Board, BoardDTO> fn = (entity -> entityToDto(entity));  
    return new PageResultDTO<>(result, fn);  
}
```

Board Project

➤ 게시판 목록 처리 (BoardService 처리 테스트)

```
@Test
public void getListTest(){
    PageRequestDTO requestDTO = PageRequestDTO.builder().page(1).size(10).build();
    PageResultDTO<BoardDTO, Board> resultDTO = boardService.getList(requestDTO);
    for(BoardDTO dto : resultDTO.getDtoList()){
        System.out.println(dto);
    }
    int totalPages = resultDTO.getTotalPage(); //전체페이지
    int startPage = resultDTO.getStart(); //화면 시작페이지
    int endPage = resultDTO.getEnd(); //화면 끝페이지
    boolean prev = resultDTO.isPrev(); //이전
    boolean next = resultDTO.isNext();
    System.out.println("totalPage:" + totalPages);
    System.out.println("startPage:" + startPage);
    System.out.println("endPage:" + endPage);
    System.out.println("prev:" + prev);
    System.out.println("next:" + next);
    for(int i : resultDTO.getPageList()){
        System.out.println("page list num:" + i);
    }
}
```

```
BoardDTO(id=101, title=Test Title, content=Test Content, hit=0, regDate=2023-05-11T14:59:37, modDate=null, writer=test, email=test@gmail.com)
BoardDTO(id=100, title=Title100, content=Content 100, hit=0, regDate=2023-05-11T13:01:31, modDate=null, writer=user2, email=user2@gmail.com)
BoardDTO(id=99, title=Title99, content=Content 99, hit=0, regDate=2023-05-11T13:01:30, modDate=null, writer=user1, email=user1@gmail.com)
BoardDTO(id=98, title=Title98, content=Content 98, hit=0, regDate=2023-05-11T13:01:30, modDate=null, writer=user6, email=user6@gmail.com)
BoardDTO(id=97, title=Title97, content=Content 97, hit=0, regDate=2023-05-11T13:01:30, modDate=null, writer=user3, email=user3@gmail.com)
BoardDTO(id=96, title=Title96, content=Content 96, hit=0, regDate=2023-05-11T13:01:30, modDate=null, writer=user8, email=user8@gmail.com)
BoardDTO(id=95, title=Title95, content=Content 95, hit=0, regDate=2023-05-11T13:01:30, modDate=null, writer=user1, email=user1@gmail.com)
BoardDTO(id=94, title=Title94, content=Content 94, hit=0, regDate=2023-05-11T13:01:30, modDate=null, writer=user1, email=user1@gmail.com)
BoardDTO(id=93, title=Title93, content=Content 93, hit=0, regDate=2023-05-11T13:01:30, modDate=null, writer=user3, email=user3@gmail.com)
BoardDTO(id=92, title=Title92, content=Content 92, hit=0, regDate=2023-05-11T13:01:30, modDate=null, writer=user9, email=user9@gmail.com)
```

```
totalPage:11
startPage:1
endPage:10
prev:false
next:true
```

```
page list num:1
page list num:2
page list num:3
page list num:4
page list num:5
page list num:6
page list num:7
page list num:8
page list num:9
page list num:10
```

Board Project

➤ 게시판 목록 처리 (컨트롤러 '/list' API 추가)

```
@Slf4j
@Controller
public class BoardController {

    private BoardService boardService;

    @Autowired
    public BoardController(BoardService boardService){
        this.boardService = boardService;
    }

    @GetMapping("/")
    public String main(){
        log.info("list");
        return "redirect:/list";
    }

    @GetMapping("/list")
    public String list(PageRequestDTO pageRequestDTO, Model model){
        log.info("list pageRequestDTO:{}", pageRequestDTO);
        PageResultDTO<BoardDTO, Board> result = boardService.getList(pageRequestDTO);
        model.addAttribute("result", result);
        return "/list";
    }
}
```

서비스 의존성 주입

목록 API 추가

Board Project

- 게시판 목록 처리 (목록화면 구성, 부트스트랩 이용)
 - ✓ form, select, table, pagination 검색

Logo

[로그인](#) [로그아웃](#)

게시판 목록

전체 ▾

검색어 입력

검색

등록

#	타이틀	본문	작성자	등록일
1	Mark	Otto	@mdo	@mdo

Previous

1

2

3

Next

Board Project

➤ 게시판 목록 처리 (목록화면 구성, 부트스트랩 이용)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>목록</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" >
</head>
<body>
<div class="container">
<nav class="navbar">
  <a class="navbar-brand">Logo</a>
  <div class="d-flex">
    <a class="btn-primary m-1" href="/login">로그인</a>
    <a class="btn-primary m-1" href="/logout">로그아웃</a>
  </div>
</nav>

  <h1>게시판 목록</h1>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

부트스트랩 기본 화면 구성

로그인/로그아웃 추가

Board Project

➤ 게시판 목록 처리 (목록화면 구성, 부트스트랩 이용)

```
<form class="row g-3" method="get" action="/list">
  <input type="hidden" name="page" value="1">
  <div class="col-auto">
    <select class="form-select" id="type" name="type">
      <option value="a" selected>전체 </option>
      <option value="t">타이틀 </option>
      <option value="c">본문 </option>
      <option value="w">작성자 </option>
    </select>
  </div>
  <div class="col-auto">
    <label for="keyword" class="visually-hidden">검색어 </label>
    <input type="text" class="form-control" id="keyword" name="keyword" placeholder="검색어 입력">
  </div>
  <div class="col-auto">
    <button type="submit" class="btn btn-primary mb-3">검색 </button>
    <button type="button" class="btn btn-primary mb-3">등록 </button>
  </div>
</form>
```

검색 영역

Board Project

➤ 게시판 목록 처리 (목록화면 구성, 부트스트랩 이용)

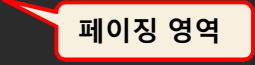
```
<table class="table">
  <thead class="table-light">
    <tr>
      <th scope="col">#</th>
      <th scope="col">타이틀</th>
      <th scope="col">본문</th>
      <th scope="col">작성자</th>
      <th scope="col">등록일</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Otto</td>
      <td>@mdo</td>
      <td>@mdo</td>
    </tr>
  </tbody>
</table>
```

리스트 테이블 영역

Board Project

➤ 게시판 목록 처리 (목록화면 구성, 부트스트랩 이용)

```
<ul class="pagination">  
  <li class="page-item disabled">  
    <span class="page-link">Previous</span>  
  </li>  
  <li class="page-item"><a class="page-link" href="#">1</a></li>  
  <li class="page-item active" aria-current="page">  
    <span class="page-link">2</span>  
  </li>  
  <li class="page-item"><a class="page-link" href="#">3</a></li>  
  <li class="page-item">  
    <a class="page-link" href="#">Next</a>  
  </li>  
</ul>
```



Board Project

➤ 게시판 목록 처리 (검색 기능 서비스 API 추가)

```
@Override
public PageResultDTO<BoardDTO, Board> getListV2(PageRequestDTO requestDTO){
    Pageable pageable = requestDTO.getPageable(Sort.by("id").descending());

    Page<Board> result = boardRepository.findAll(getSearch(requestDTO), pageable);

    Function<Board, BoardDTO> fn = (entity -> entityToDto(entity));
    return new PageResultDTO<>(result, fn);
}
```

Board Project

➤ 게시판 목록 처리 (검색 기능 서비스 API 추가)

```
public BooleanBuilder getSearch(PageRequestDTO requestDto){
    String type = requestDto.getType();
    type = type!=null&&type.isEmpty()==false ? type : "a";

    //엔티티 검색
    QBoard qBoard = QBoard.board;
    String keyword = requestDto.getKeyword();
    keyword = keyword!=null&&keyword.isEmpty()==false ? keyword : "";
    BooleanBuilder builder = new BooleanBuilder();
    switch (type){
        case "a": {
            BooleanExpression allExp = qBoard.title.contains(keyword);
            BooleanExpression contentExp = qBoard.content.contains(keyword);
            BooleanExpression writerExp = qBoard.member.writer.contains(keyword);
            builder.and(allExp.or(contentExp).or(writerExp));
            break;
        }
    }
    ... 생략 ...
}
```

Board Project

➤ 게시판 목록 처리 (검색 기능 서비스 API 추가)

```
... 생략 ...
    case "t":
        BooleanExpression titleExp = qBoard.title.contains(keyword);
        builder.and(titleExp);
        break;
    case "c":
        BooleanExpression contentExp = qBoard.content.contains(keyword);
        builder.and(contentExp);
        break;
    case "w":
        BooleanExpression writerExp = qBoard.member.writer.contains(keyword);
        builder.and(writerExp);
        break;
}
builder.and(qBoard.id.gt(0L));
return builder;
}
```

Board Project

➤ 게시판 목록 처리 (컨트롤러 검색기능 '/list' API 변경)

```
@GetMapping("/list")
public String list(PageRequestDTO pageRequestDTO, Model model){
    log.info("list pageRequestDTO:{}, pageRequestDTO);

    //검색어 디폴트
    PageRequestDTO requestDTO = pageRequestDTO;
    requestDTO.setType(requestDTO.getType()==null?"a":requestDTO.getType());

    PageResultDTO<BoardDTO, Board> result = boardService.getListV2(pageRequestDTO);
    model.addAttribute("result", result);
    model.addAttribute("request", requestDTO);
    return "/list";
}
```

Board Project

➤ 게시판 목록 처리 (화면 수정 : 데이터 연동)

```
<tr th:each="item, status : ${result.dtoList}">
  <th scope="row" th:text="${status.count}">1</th>
  <td th:text="${item.title}">Mark</td>
  <td th:text="${item.content}">Otto</td>
  <td th:text="${item.writer}">@mdo</td>
  <td th:text="${#temporals.format(item.regDate, 'yyyy/MM/dd')}">@mdo</td>
</tr>
```

Board Project

➤ 게시판 목록 처리 (화면 수정 : 페이징 처리 연동)

```
<ul class="pagination justify-content-center align-items-center">
  <li class="page-item" th:if="${result.prev}">
    <a class="page-link" th:href="@{/list(page=${result.start-1}, type=${request.type}, keyword=${request.keyword})}">Previous</a>
  </li>

  <li th:class="${result.page==page?'page-item active':'page-item'}" th:each="page : ${result.pageList}">
    <a class="page-link" th:href="@{/list(page=${page}, type=${request.type}, keyword=${request.keyword})}" th:text="${page}">3</a>
  </li>

  <li class="page-item" th:if="${result.next}">
    <a class="page-link" th:href="@{/list(page=${result.end+1}, type=${request.type}, keyword=${request.keyword})}">Next</a>
  </li>
</ul>
```

Board Project

➤ 게시판 목록 처리

```
<nav class="navbar">
  <a class="navbar-brand">Logo</a>
  <div class="d-flex">
    <th:block th:if="${session.member != null}">
      <a class="btn-primary m-1" href="/mypage" th:text="${session.member?.name}" ></a>
      <a class="btn-primary m-1" href="/logout" >로그아웃</a>
    </th:block>
    <th:block th:unless="${session.member != null}">
      <a class="btn-primary m-1" href="/login" >로그인</a>
    </th:block>
  </div>
</nav>
```

로그인 세션 유무에 따른
로그인/로그아웃 메뉴 처리

Board Project

➤ 게시판 목록 처리 결과 화면

게시판 목록				
<div>전체 ▼ 검색어 입력 검색 등록</div>				
#	타이틀	본문	작성자	등록일
1	Test Title	Test Content	test	2023/05/11
2	Title100	Content 100	user2	2023/05/11
3	Title99	Content 99	user1	2023/05/11
4	Title98	Content 98	user6	2023/05/11
5	Title97	Content 97	user3	2023/05/11
6	Title96	Content 96	user8	2023/05/11
7	Title95	Content 95	user1	2023/05/11
8	Title94	Content 94	user1	2023/05/11
9	Title93	Content 93	user3	2023/05/11
10	Title92	Content 92	user9	2023/05/11
<div>1 2 3 4 5 6 7 8 9 10 Next</div>				

Board Project

- 게시판 신규 등록 처리(등록화면 구성, 부트스트랩 이용)
 - ✓ form 검색

게시판 등록

작성자

타이틀

작성내용

저장취소

Board Project

➤ 게시판 신규 등록 처리(등록화면 구성, 부트스트랩 이용)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>신규등록</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css"
rel="stylesheet" >
</head>
<body>
  <div class="container">
    <h1>게시판 등록</h1>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" ></script>
</body>
</html>
```



부트스트랩 기본 화면 구성

Board Project

➤ 게시판 신규 등록 처리(등록화면 구성, 부트스트랩 이용)

```
<div class="container">
  <h1>게시판 등록</h1>
  <form action="/write" method="post">
    <div class="form-group col-3">
      <label for="writer" class="form-label">작성자</label>
      <input type="text" class="form-control" id="writer" name="writer" placeholder="작성자" readonly >
    </div>
    <div class="form-group col-6">
      <label for="title" class="form-label">타이틀</label>
      <input type="text" class="form-control" id="title" name="title" placeholder="타이틀">
    </div>
    <div class="form-group col-6">
      <label for="content" class="form-label">작성내용</label>
      <textarea class="form-control" id="content" name="content" rows="3"></textarea>
    </div>
    <button type="submit" class="btn btn-primary mt-3">저장</button>
    <button type="button" class="btn btn-secondary mt-3">취소</button>
  </form>
</div>
```

```
log.info("write");
if(member==null){
    return "redirect:/login";
}
return "/write";
}
```

➤ 게시판 신규 등록 처리

- ✓ 컨트롤러 '/write' GET API 추가

```
//신규생성 화면
@GetMapping("/write")
public String write(@SessionAttribute(name = "member", required = false) MemberDTO member){
    log.info("write");
    if(member==null){
        return "redirect:/login";
    }
    return "/write";
}
```

로그인 세션이 없으면
로그인 창으로 이동 처리

- ✓ 신규 등록화면 > 작성자 정보 > 로그인 세션 정보에서 조회

```
<div class="form-group col-3">
    <label for="writer" class="form-label">작성자</label>
    <input type="text" class="form-control" id="writer" name="writer" placeholder="작성자" readonly
        th:value="${session.member?.name}" >
</div>
```

로그인 세션 정보

Board Project

- 게시판 신규 등록 처리
 - ✓ 컨트롤러 '/write' POST API 추가

```
@PostMapping("/write")
public String doWrite(BoardDTO dto, RedirectAttributes redirectAttributes
    , @SessionAttribute(name = "member", required = false) MemberDTO member){
    log.info("doWrite dto:" + dto);
    if(member==null){
        return "redirect:/login";
    }
    //로그인 세션정보로 작성자 주입
    dto.setEmail(member.getEmail());
    Long id = boardService.register(dto);
    if(id!=null && id>0L){
        redirectAttributes.addFlashAttribute("msg", "등록 되었습니다");
    }
    return "redirect:/list";
}
```

작성 정보 저장

로그인 세션이 없으면
로그인 창으로 이동 처리

작성자 정보 로그인 세션
정보에서 주입

Board Project

➤ 게시판 상세화면 처리

게시판 상세화면

작성자

타이틀

작성내용

등록일

조회수

목록

수정

삭제

Board Project

➤ 게시판 상세화면 처리

```
<div class="container">
  <h1>게시판 상세화면</h1>
  <div class="mb-3 col-3">
    <label for="writer" class="form-label">작성자</label>
    <input type="text" class="form-control" id="writer" name="writer" placeholder="작성자" readonly>
  </div>
  <div class="mb-3 col-6">
    <label for="title" class="form-label">타이틀</label>
    <input type="text" class="form-control" id="title" name="title" placeholder="타이틀" readonly>
  </div>
  <div class="mb-3 col-6">
    <label for="content" class="form-label">작성내용</label>
    <textarea class="form-control" id="content" name="content" rows="3" readonly></textarea>
  </div>
  <div class="mb-3 col-3">
    <label for="regDate" class="form-label">등록일</label>
    <input type="text" class="form-control" id="regDate" name="regDate" readonly>
  </div>
  <div class="mb-3 col-3">
    <label for="hit" class="form-label">조회수</label>
    <input type="text" class="form-control" id="hit" name="hit" readonly>
  </div>
  <button type="submit" class="btn btn-secondary mt-3">목록</button>
  <button type="button" class="btn btn-primary mt-3">수정</button>
  <button type="button" class="btn btn-danger mt-3">삭제</button>
</div>
```

읽기 전용 처리

목록, 수정, 삭제
버튼 추가

Board Project

- 게시판 상세화면 처리
 - ✓ 서비스 상세조회 API 추가

```
//상세조회  
BoardDTO findById(Long id, boolean hit);
```

```
@Override public BoardDTO findById(Long id, boolean hit) {  
    Optional<Board> boardOptional = boardRepository.findById(id);  
    if(boardOptional.isPresent()){  
        Board board = boardOptional.get();  
        if(hit){  
            board.setHit(board.getHit()+1); //조회수 추가  
            board = boardRepository.save(board);  
        }  
        return entityToDto(board);  
    }  
    return null;  
}
```

Board Project

- 게시판 상세화면 처리
 - ✓ 컨트롤러 '/detail' API 추가

```
@GetMapping("/detail/{id}")
public String detail(@PathVariable Long id, Model model){
    log.info("detail");
    BoardDTO boardDTO = boardService.findById(id, true);
    if(boardDTO==null){
        return "redirect:/list";
    }
    model.addAttribute("dto", boardDTO);
    return "/detail";
}
```

Board Project

- 게시판 상세화면 처리
 - ✓ 목록화면 > 상세화면 호출 추가

```
<tbody>
<tr th:each="item, status : ${result.dtoList}" th:onclick="location.href='/detail/[[${item.id}]]'">
  <th scope="row" th:text="${status.count}">1</th>
  <td th:text="${item.title}">Mark</td>
  <td th:text="${item.content}">Otto</td>
  <td th:text="${item.writer}">@mdo</td>
  <td th:text="${#temporals.format(item.regDate, 'yyyy/MM/dd')}">@mdo</td>
</tr>
</tbody>
```

Board Project

➤ 게시판 상세화면 처리

✓ 상세화면 데이터 연동

```
<div class="container">
  <h1>게시판 상세화면</h1>
  <div class="mb-3 col-3">
    <label for="writer" class="form-label">작성자</label>
    <input type="text" class="form-control" id="writer" name="writer" placeholder="작성자" th:value="${dto.writer}" readonly>
  </div>
  <div class="mb-3 col-6">
    <label for="title" class="form-label">타이틀</label>
    <input type="text" class="form-control" id="title" name="title" placeholder="타이틀" th:value="${dto.title}" readonly>
  </div>
  <div class="mb-3 col-6">
    <label for="content" class="form-label">작성내용</label>
    <textarea class="form-control" id="content" name="content" rows="3" readonly>[[${dto.content}]]</textarea>
  </div>
  <div class="mb-3 col-3">
    <label for="regDate" class="form-label">등록일</label>
    <input type="text" class="form-control" id="regDate" name="regDate" th:value="${#temporals.format(dto.regDate, 'yyyy/MM/dd HH:mm:ss')}" readonly>
  </div>
  <button type="submit" class="btn btn-secondary mt-3" th:onclick="location.href='/list'">목록</button>
  <th:block th:if="${session.member?.email == dto.email}">
    <button type="button" class="btn btn-primary mt-3" th:onclick="location.href='/modify/[[${dto.id}]]'">수정</button>
    <button type="button" class="btn btn-danger mt-3" th:onclick="del([[${dto.id}]])">삭제</button>
  </th:block>
</div>
```

데이터 연결

수정, 삭제 버튼은
본인 작성글에만 표시되도록 처리

Board Project

➤ 게시판 수정화면 처리

게시판 수정

작성자

작성자

읽기 전용 처리

타이틀

타이틀

타이틀, 작성내용만
수정 처리

작성내용

수정 취소

Board Project

➤ 게시판 수정화면 처리

```
<div class="container">
  <h1>게시판 수정</h1>
  <form action="/modify" method="post">
    <input type="hidden" name="id" th:value="${dto.id}">
    <div class="mb-3 col-3">
      <label for="writer" class="form-label">작성자</label>
      <input type="text" class="form-control" id="writer" name="writer" placeholder="작성자" readonly>
    </div>
    <div class="mb-3 col-6">
      <label for="title" class="form-label">타이틀</label>
      <input type="text" class="form-control" id="title" name="title" placeholder="타이틀">
    </div>
    <div class="mb-3 col-6">
      <label for="content" class="form-label">작성내용</label>
      <textarea class="form-control" id="content" name="content" rows="3"></textarea>
    </div>
    <button type="submit" class="btn btn-primary mt-3">수정</button>
    <button type="button" class="btn btn-secondary mt-3">취소</button>
  </form>
</div>
```

읽기 전용 처리

Board Project

- 게시판 수정화면 처리
 - ✓ 서비스 업데이트 API 추가

```
//수정
BoardDTO update(BoardDTO dto);
```

```
@Override public BoardDTO update(BoardDTO dto) {
    Optional<Board> boardOptional = boardRepository.findById(dto.getId());
    if(boardOptional.isPresent()){
        Board entity = boardOptional.get();
        //수정항목만 업데이트
        entity.setTitle(dto.getTitle());
        entity.setContent(dto.getContent());
        entity = boardRepository.save(entity);
        return entityToDto(entity);
    }
    return null;
}
```

Board Project

- 게시판 수정화면 처리
 - ✓ 컨트롤러 '/modify' GET API 추가

```
//수정 화면
@GetMapping("/modify/{id}")
public String modify(@PathVariable Long id, Model model){
    log.info("modify");
    BoardDTO boardDTO = boardService.findById(id, false);
    if(boardDTO==null){
        return "redirect:/list";
    }
    model.addAttribute("dto", boardDTO);
    return "/modify";
}
```


Board Project

- 게시판 수정화면 처리
 - ✓ 컨트롤러 '/modify' POST API 추가

```
//수정
@PostMapping("/modify")
public String update(BoardDTO dto){
    log.info("update dto:{}", dto);
    BoardDTO boardDTO = boardService.update(dto);
    if(boardDTO==null || boardDTO.getId()==null){
        return "redirect:/list";
    }
    return "redirect:/detail/"+dto.getId();
}
```

Board Project

➤ 게시판 삭제 처리

게시판 상세화면

작성자

타이틀

작성내용

등록일

목록 수정 삭제 삭제 처리

localhost:8080 내용:

삭제 하시겠습니까?

확인 취소

Board Project

➤ 게시판 삭제 처리(detail.html 파일 수정)

```
<script th:inline="javascript">
  function del(id) {
    if(!confirm('삭제 하시겠습니까?')){
      return;
    }
    let jsonObj = {'id':id};
    let jsonStr = JSON.stringify(jsonObj)
    $.ajax({
      type: "POST",
      url: "/remove",
      contentType: "application/json; charset=utf-8",
      data:jsonStr,
      success: function(data, status, xhr) {
        if(data!=null && data!=undefined && data.result==1){
          alert(data.msg);
          window.location.replace('/list');
        }else{
          alert('삭제를 하지 못했습니다');
        }
      },
      error: function(xhr, status, error) {
        alert('삭제를 하지 못했습니다');
      }
    });
  }
</script>
```

삭제 처리 Javascript ajax API 추가

삭제 처리 Javascript ajax API 추가

Board Project

➤ 게시판 삭제 처리

- ✓ 서비스 삭제 API 추가

```
//삭제  
int remove(Long id);
```

```
@Override public int remove(Long id) {  
    int result = 0;  
    try{  
        boardRepository.deleteById(id);  
        result = 1;  
    }catch (Exception e){  
        e.printStackTrace();  
    }  
    return result;  
}
```

Board Project

➤ 게시판 삭제 처리

✓ 컨트롤러 '/remove' API 추가

```
@PostMapping("/remove")
@ResponseBody
public ResponseEntity<Object> remove(@RequestBody Map<String, Object> map){
    log.info("remove map:{}", map);
    int tempId = (int) map.get("id");
    Long id = Long.valueOf(tempId);
    if(id==null){
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("삭제를 하지 못했습니다.");
    }

    int result = boardService.remove(id);
    HashMap res = new HashMap();
    res.put("result", result);
    if(result>0){
        res.put("msg", "삭제가 되었습니다.");
    }else{
        res.put("msg", "삭제를 하지 못했습니다.");
    }
    return ResponseEntity.status(HttpStatus.OK).body(res);
}
```