이벤트 구현 실습을 하려고 한다.

schema.sql – 테이블 생성

```
DROP TABLE IF EXISTS comment;
DROP TABLE IF EXISTS post;

CREATE TABLE post (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    content TEXT
);

CREATE TABLE comment (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    post_id BIGINT NOT NULL,
    writer VARCHAR(100),
    content TEXT,
    FOREIGN KEY (post_id) REFERENCES post(id) ON DELETE CASCADE
);
```

data.sql – 테스트 데이터 삽입

```
INSERT INTO post (title, content) VALUES
('Spring 이벤트 게시판', '이벤트 기반 구조를 연습합니다.'),
('두 번째 게시글', '댓글 기능을 테스트할 예정입니다.');

INSERT INTO comment (post_id, writer, content) VALUES
(1, 'kim', '좋은 글이에요!'),
(1, 'lee', '비동기 이벤트 테스트 중입니다.'),
(2, 'park', '삭제 이벤트가 작동할까요?');
```

application.yml 설정

```
spring:
  datasource:
    url: jdbc:h2:mem:testdb
    driver-class-name: org.h2.Driver
    username: sa
    password:
  h2:
    console:
      enabled: true
  sql:
    init:
      mode: always
```

사용법

1. 위 schema.sql, data.sql 파일을 src/main/resources/에 넣으세요.
2. 애플리케이션 실행 시 자동으로 테이블이 생성되고, 테스트 데이터가 삽입됩니다.
3. H2 콘솔로 접속 시 쿼리 확인 가능:
    - URL: jdbc:h2:mem:testdb
    - 테이블: POST, COMMENT

[문제 1] 게시글 등록 시 동기 이벤트로 알림 로그 출력하기

요구사항에 따라 게시글 등록 이벤트를 발행하고 처리하시오.

항목	내용
이벤트 클래스	PostCreatedEvent 클래스를 작성한다. 필드: postId, title
서비스 수정	PostService.createPost() 안에서 이벤트를 발행한다 (ApplicationEventPublisher)
리스너 구현	@EventListener 로 동기 이벤트 리스너를 작성하여 System.out.println("게시글 등록됨: "+ title) 출력

디렉토리 구조

```
src/
main/ java/com/
        └── board/
            ├── entity/
            |   └── Post.java                |
            ├── event/
            |   └── PostCreatedEvent.java
            ├── listener/
            |   └── PostEventListener.java
            ├── service/
            |   └── PostService.java
            ├── controller/
            |   └── PostController.java
            └── SpringBootLabEventBoardApplication.java
    └── resources/
        └── application.yml
```

테스트 방법

1. PostService.createPost(String title) 내부에 다음 코드 추가

```
System.out.println("서비스 내부 로직 실행 완료");
publisher.publishEvent(new PostCreatedEvent(postId, title));
System.out.println("이벤트 발행 후 로직 계속 진행됨");
```

2. PostEventListener 의 @EventListener 메서드에 다음 코드 작성

```
System.out.println("이벤트 리스너 실행됨: " + event.getTitle());
```

3. PostController 에 /posts POST 요청을 처리하는 메서드 작성

메서드명	리턴값	HTTP 메서드	파라미터	설명
create	Post	POST /posts	@RequestBody Post post	게시글 등록 + PostCreatedEvent 동기 발행
all	List<Post>	GET /posts	없음	전체 게시글 목록 조회
delete	void	DELETE /posts/{id}	@PathVariable Long id	게시글 삭제 + PostDeletedEvent 발행

테스트 포인트

- 로그 순서로 동기 흐름 확인
- 이벤트 → 리스너 → 다음 코드 순서로 출력되면 성공

[문제 2] 댓글 등록 시 비동기 이벤트로 외부 알림 흉내내기

댓글이 작성되면 비동기 이벤트로 알림을 전송하는 구조를 구현하시오.

문제 1 번의 내용으로 확장

항목	내용
이벤트 클래스	CommentCreatedEvent: postId, writer, content 포함
서비스 수정	CommentService.addComment() 내에서 CommentCreatedEvent 발행
비동기 리스너	@Async + @EventListener 로 별도 클래스에서 리스닝 System.out.println("새 댓글 알림 전송 중...") 출력
설정	@EnableAsync 필수 설정 추가
목적	비동기 이벤트 흐름, 지연 처리, 성능 분리 체험

디렉토리구조

```
src/
main/ java/com/
                └─── board/
                ├─── entity/
                |    └─── Post.java                    # 게시글 엔티티
                ├─── event/
                |    └─── PostCreatedEvent.java        # 게시글 등록 이벤트
                ├─── listener/
                |    └─── PostEventListener.java       # 이벤트 리스너 (동기)
                ├─── service/
                |    └─── PostService.java             # 게시글 저장 + 이벤트 발행
                ├─── controller/
                |    └─── PostController.java          # POST /posts API
                └─── SpringBootLabEventBoardApplication.java
```

테스트 방법

1. CommentService.addComment()코드에서

   ```
   publisher.publishEvent(new CommentCreatedEvent(postId, writer,
   content));
   System.out.println("비동기 이벤트 발행 완료");
   ```

2. CommentEventListener 코드에서

   ```
   @Async
   @EventListener
   public void handle(CommentCreatedEvent event) {
       Thread.sleep(2000);  // 일부러 지연
       System.out.println("댓글 이벤트 처리 중... by " +
   event.getWriter());
   }
   ```

3. AsyncConfig.java 작성 후 @EnableAsync 적용 확인
4. CommentController

메서드명	리턴값	HTTP 메서드	파라미터	설명
add	Comment	POST /comments/{postId}	@PathVariable Long postId, @RequestBody Comment comment	댓글 등록 + CommentCreatedEvent 비동기 발행

테스트 포인트

- 비동기 실행 로그가 **나중에 출력**되는지 확인

```
비동기 이벤트 발행 완료

(2초 후) 댓글 이벤트 처리 중... by 홍길동
```

[문제 3] 게시글 삭제 시 관련 댓글 삭제를 커스텀 이벤트로 처리

게시글 삭제 시 관련 댓글을 함께 삭제하는 로직을 커스텀 이벤트로 분리하시오.

항목	내용
이벤트 클래스	PostDeletedEvent: postId 포함
서비스 로직	PostService.deletePost() 내부에서 PostDeletedEvent 를 발행하고, 실제 댓글 삭제는 리스너에서 처리
리스너 처리	@EventListener 또는 @TransactionalEventListener 로 CommentRepository.deleteByPostId(postId) 실행
장점	도메인 분리, 결합도 낮춤, 테스트 용이성 향상
확장	향후 댓글 외에도 첨부파일, 좋아요 등 후처리를 분리하기 쉬움

디렉토리 구조

```
src/ main/ java/ com/
        └─── board/
                ├─── entity/
                |    ├─── Post.java
                |    └─── Comment.java
                ├─── event/
                |    ├─── PostCreatedEvent.java
                |    ├─── CommentCreatedEvent.java
                |    └─── PostDeletedEvent.java          # 게시글 삭제 이벤트
                ├─── listener/
                |    ├─── PostEventListener.java
                |    └─── CommentEventListener.java      # PostDeletedEvent 처리
                ├─── service/
                |    ├─── PostService.java               # 삭제 후 PostDeletedEvent 발행
                |    └─── CommentService.java
                ├─── controller/
                |    ├─── PostController.java            # DELETE /posts/{id}
                |    └─── CommentController.java
                └─── SpringBootLabEventBoardApplication.java
```

테스트 방법

1. PostService.deletePost(Long id) 코드

   ```
   publisher.publishEvent(new PostDeletedEvent(id));
   ```

2. CommentEventListener 또는 별도 PostDeleteListener 코드

   ```
   @EventListener
   public void onPostDeleted(PostDeletedEvent event) {
       System.out.println("해당 게시글의 댓글 삭제 실행: postId = " +
   event.getPostId());
       commentRepository.deleteByPostId(event.getPostId());
   }
   ```

3. PostController

메서드명	리턴값	HTTP 메서드	파라미터	설명
delete	void	DELETE /posts/{id}	@PathVariable Long id	커스텀 POJO 이벤트(PostDeletedEvent)로 댓글 삭제 트리거

테스트 포인트

- 실제 댓글 삭제 쿼리 실행 여부는 콘솔 로그 또는 H2 콘솔로 확인 가능
- 삭제 후 연관 댓글이 제거되는지 검증

테스트 유틸

방법	설명
Postman / REST Client	POST /posts, POST /comments, DELETE /posts/{id} 호출 시도
H2 콘솔	spring.h2.console.enabled=true 설정 후 DB 테이블 확인
로그 확인	System.out.println() 또는 Logger 사용하여 이벤트 흐름 추적