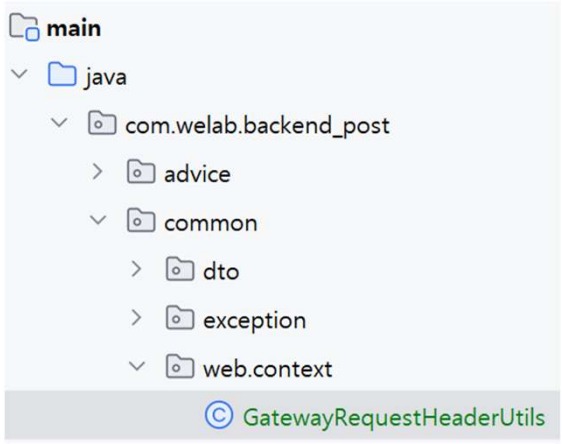
서비스 로직 작성

Post Service 기능 구현

Post 프로젝트 코드 작성 – GatewayRequestHeaderUtils

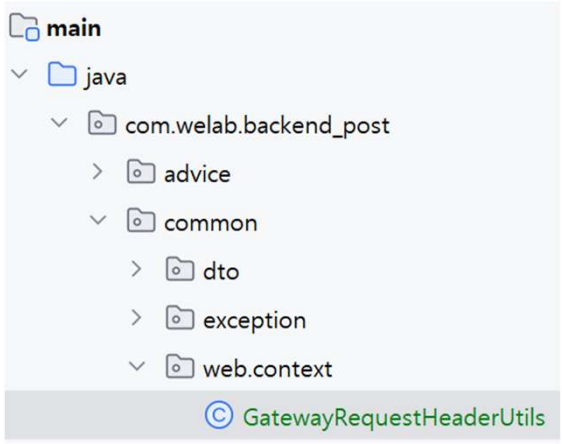



```
public class GatewayRequestHeaderUtils {
    public static String getRequestHeaderParamAsString(String key) {
        ServletRequestAttributes requestAttributes =
                (ServletRequestAttributes) RequestContextHolder.currentRequestAttributes();
        return requestAttributes.getRequest().getHeader(key);
    }
```

API Gateway에서 추가한 헤더 값을 추출하는 Util 함수

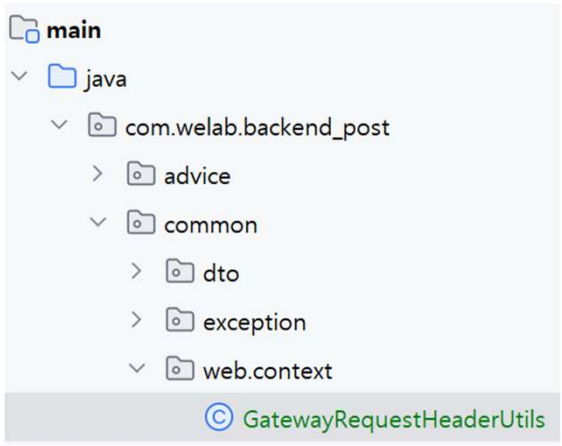
User 서비스와 동일하게 구성

Post 프로젝트 코드 작성 – GatewayRequestHeaderUtils (계속)



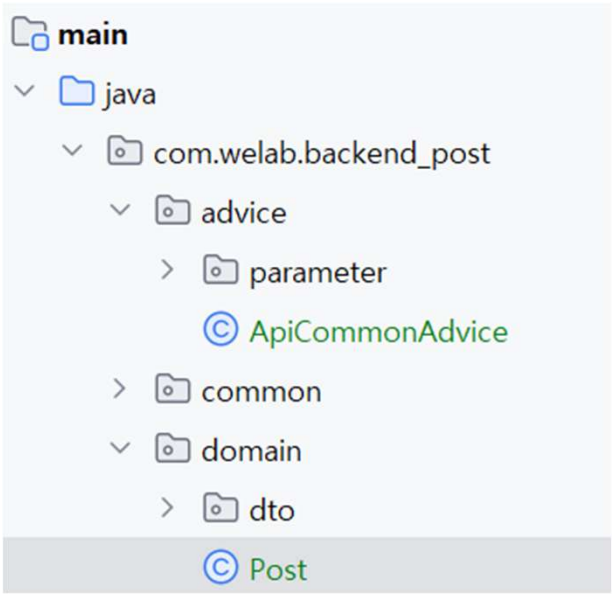

```
public static String getUserId() {
    String userId = getRequestHeaderParamAsString("X-Auth-UserId");
    if (userId == null) {
        return null;
    }

    return userId;
}

public static String getClientDevice() {
    String clientDevice = getRequestHeaderParamAsString("X-Client-Device");
    if (clientDevice == null) {
        return null;
    }

    return clientDevice;
}

public static String getClientAddress() {
    String clientAddress = getRequestHeaderParamAsString("X-Client-Address");
    if (clientAddress == null) {
        return null;
    }

    return clientAddress;
}
```

Post 프로젝트 코드 작성 – GatewayRequestHeaderUtils (계속)




```
    public static String getUserIdOrThrowException() {
        String userId = getUserId();
        if (userId == null) {
            throw new NotFound("헤더에 userId 정보가  없습니다.");
        }

        return userId;
    }

    public static String getClientDeviceOrThrowException() {
        String clientDevice = getClientDevice();
        if (clientDevice == null) {
            throw new NotFound("헤더에  사용자  디바이스  정보가  없습니다.");
        }

        return clientDevice;
    }

    public static String getClientAddressOrThrowException() {
        String clientAddress = getClientAddress();
        if (clientAddress == null) {
            throw new NotFound("헤더에  사용자 IP 주소  정보가  없습니다.");
        }

        return clientAddress;
    }
}
```

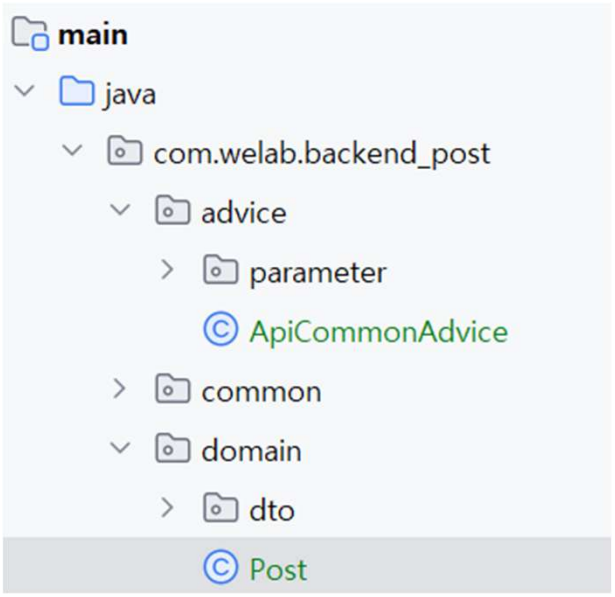
Post 프로젝트 코드 작성 – Post Entity




```java
@Slf4j
@Entity
@Table(name = "post",
        indexes = {
                @Index(columnList = "user_id"),
                @Index(columnList = "created_datetime"),
                @Index(columnList = "updated_datetime")
        })
public class Post {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter
    @Setter
    private Long id;

    @Column(name = "title", nullable = false)
    @Getter
    private String title;

    @Column(name = "content", nullable = false)
    @Getter
    private String content;
```

자주 조회하는 데이터를 인덱스로 지정
(테이블 조회를 빨리 하기 위함)

Post 프로젝트 코드 작성 – Post Entity (계속)



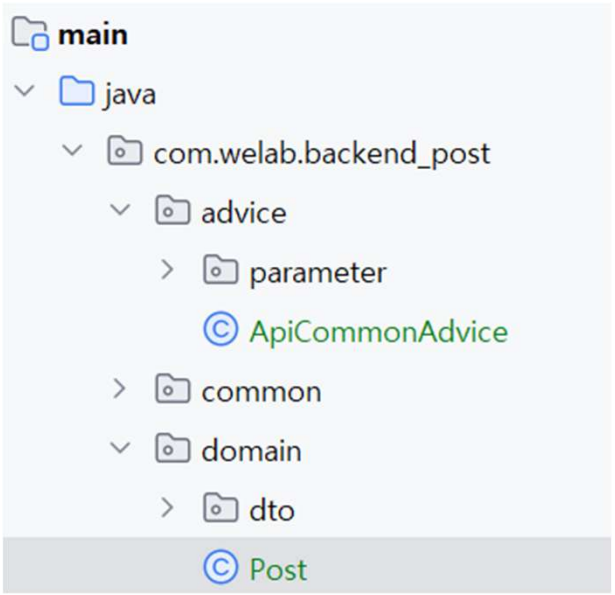
Post의 제목, 내용의 변경사항을
영속성 컨텍스트가 관리하는
엔티티에 반영하여 DB와 동기화

업데이트된 시간
= setPost 메소드가 호출된 시간

```
public void setPost(String title, String contents) {
    this.title = title;
    this.content = contents;
    this.updatedDatetime = LocalDateTime.now();
}
```

```
@Column(name = "user_id", nullable = false)
@Getter
@Setter
private String userId;

@Column(name = "created_datetime", nullable = false)
@Getter
private LocalDateTime createdDatetime = LocalDateTime.now();

@Column(name = "updated_datetime")
@Getter
private LocalDateTime updatedDatetime;
```

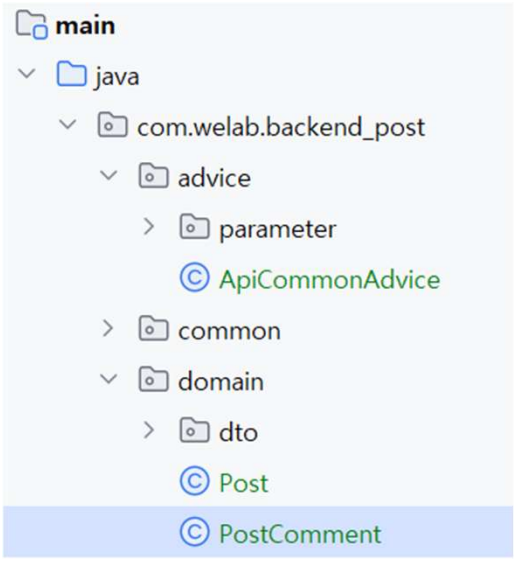
Post 프로젝트 코드 작성 – Post Entity (계속)




```
    @OneToMany(mappedBy = "post", fetch = FetchType.LAZY)
    private List<PostComment> comments = new ArrayList<>();

    public List<PostComment> getComments() {
        return this.comments;
    }

    public void addComment(PostComment comment) {
        comment.setPost(this);
        this.comments.add(comment);
    }
}
```

영속성 컨텍스트에서 DB와의 데이터 동기화를 위해 추가 (편의성 및 안정성 ↑)

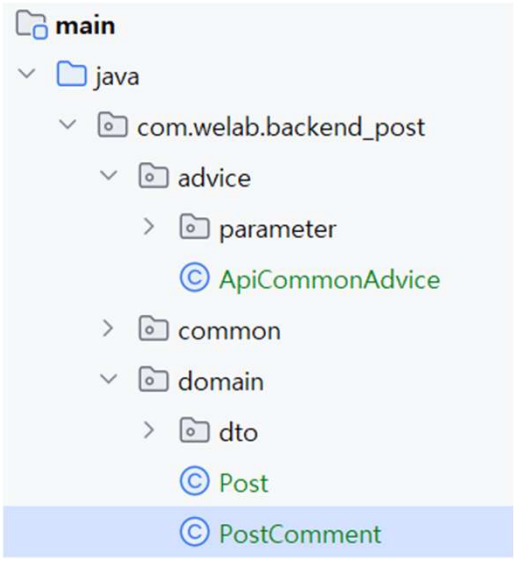
Post 프로젝트 코드 작성 – PostComment Entity




```java
@Slf4j
@Entity
@Table(name = "post_comment",
        indexes = {
                @Index(columnList = "user_id"),
                @Index(columnList = "created_datetime"),
                @Index(columnList = "updated_datetime")
        })
public class PostComment {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter
    @Setter
    private Long id;

    @Column(name = "comment", nullable = false)
    @Getter
    private String comment;

    public void setComment(String comment) {
        this.comment = comment;
        this.updatedDatetime = LocalDateTime.now();
    }
```

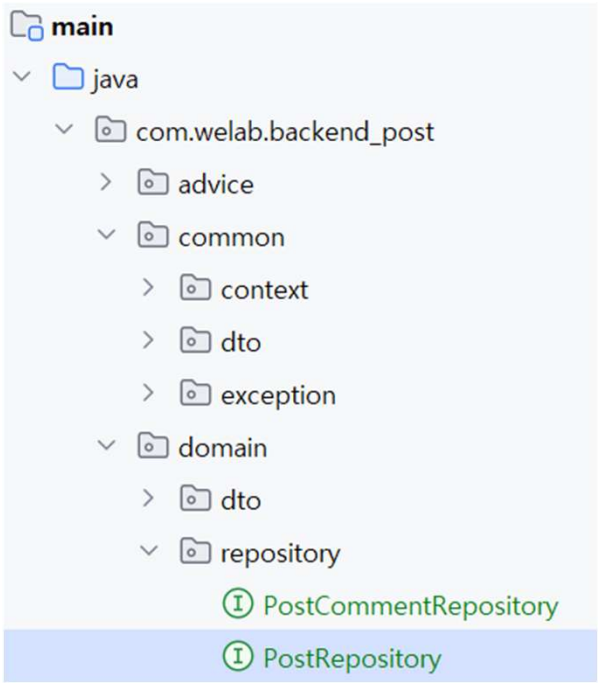
setComment() 호출 시,
업데이트 일자를 호출한 시간으로 설정

Post 프로젝트 코드 작성 – PostComment Entity (계속)



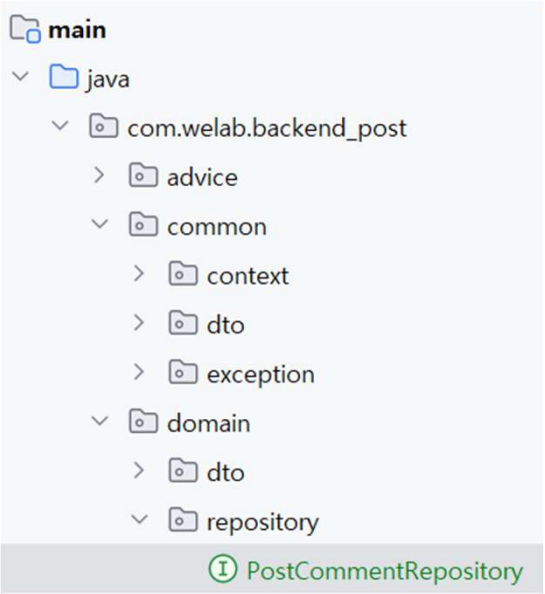

```
    @Column(name = "user_id", nullable = false)
    @Getter
    @Setter
    private String userId;

    @Column(name = "created_datetime", nullable = false)
    @Getter
    private LocalDateTime createdDatetime = LocalDateTime.now();

    @Column(name = "updated_datetime")
    @Getter
    private LocalDateTime updatedDatetime;

    @ManyToOne(targetEntity = Post.class)
    @JoinColumn(name = "post_id", referencedColumnName = "id", nullable =
true)
    @Setter
    private Post post;
}
```

Post 프로젝트 코드 작성 – PostRepository



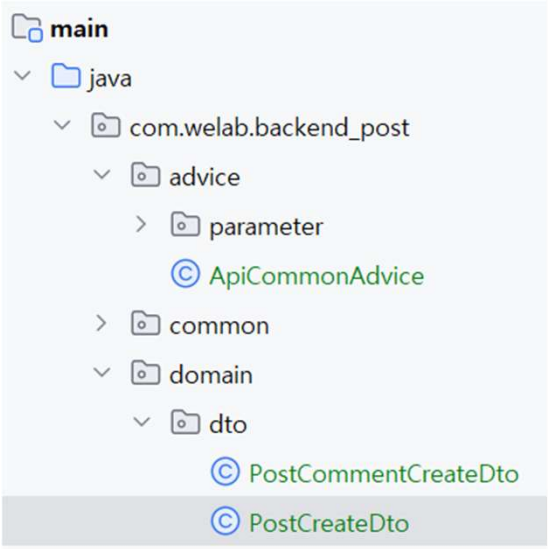

```java
@Repository
public interface PostRepository extends JpaRepository<Post, Long> {
    Post findByUserId(String userId);
}
```

Post 프로젝트 코드 작성 – PostCommentRepository




```java
@Repository
public interface PostCommentRepository extends JpaRepository<PostComment, Long> {
    PostComment findByUserId(String userId);
}
```

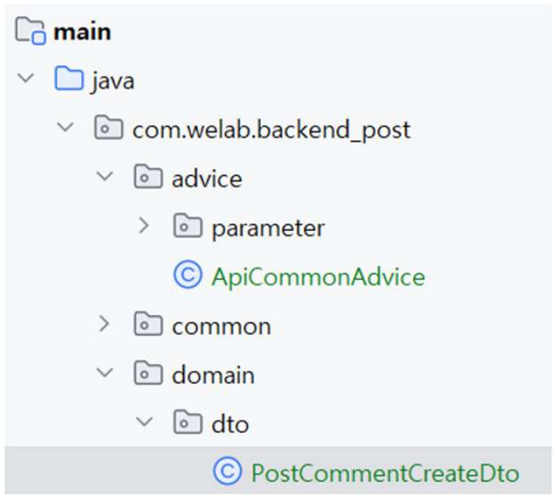
Post 프로젝트 코드 작성 – PostCreateDto




```java
@Getter
@Setter
public class PostCreateDto {
    @NotBlank(message = "타이틀을  입력하세요.")
    private String title;

    @NotBlank(message = "본문을  입력하세요.")
    private String content;

    public Post toEntity() {
        Post post = new Post();

        post.setUserId(GatewayRequestHeaderUtils.getUserIdOrThrowException());
        post.setPost(this.title, this.content);

        return post;
    }
}
```

UserId(GatewayRequestHeaderUtils 내의 함수에서 사용자 아이디 추출

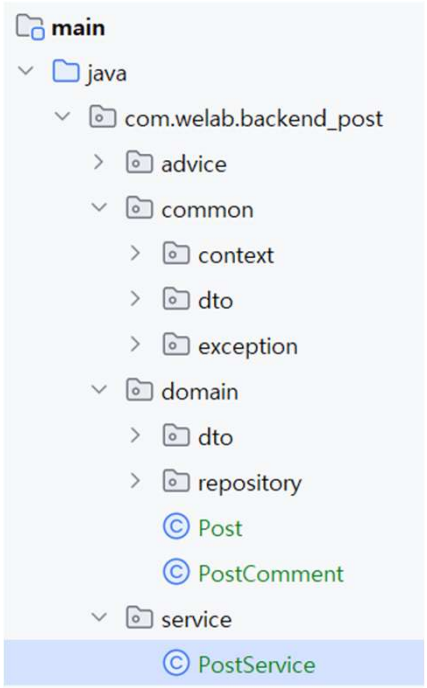
Post 프로젝트 코드 작성 – PostCommentCreateDto




```java
@Getter
@Setter
public class PostCommentCreateDto {
    @NotNull(message = "포스트 ID를  입력하세요.")
    private Long postId;

    @NotBlank(message = "댓글을  입력하세요.")
    private String comment;

    public PostComment toEntity() {
        PostComment postComment = new PostComment();

        postComment.setUserId(GatewayRequestHeaderUtils.getUserIdOrThrowException());
        postComment.setComment(this.comment);

        return postComment;
    }
}
```

UserId(GatewayRequestHeaderUtils 내의 함수에서 사용자 아이디 추출

Post 프로젝트 코드 작성 – PostService




```
@Slf4j
@Service
@RequiredArgsConstructor
public class PostService {
    private final PostRepository postRepository;
    private final PostCommentRepository postCommentRepository;

    @Transactional
    public void createPost(PostCreateDto createDto) {
        Post post = createDto.toEntity();

        postRepository.save(post);
    }

    @Transactional
    public void addPostComment(PostCommentCreateDto createDto) {
        Post post = postRepository.findById(createDto.getPostId())
                .orElseThrow(() -> new NotFound("포스팅 글을 찾을 수 없습니다."));

        PostComment postComment = createDto.toEntity();
        postCommentRepository.save(postComment);

        post.addComment(postComment);
    }
}
```

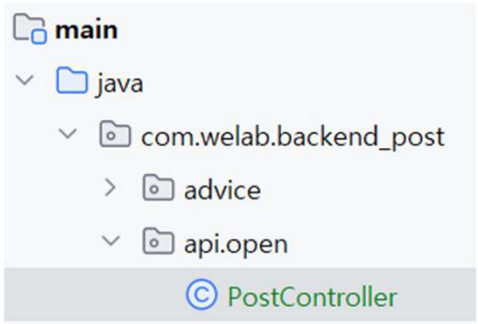
게시글 생성

댓글 작성

DB에 댓글 생성

DB에 추가된 댓글과
영속성 컨텍스트의 Post 내의
PostComment 리스트와 동기화

Post 프로젝트 코드 작성 – PostController




```java
@Slf4j
@RestController
@RequestMapping(value = "/api/post/v1", produces = MediaType.APPLICATION_JSON_VALUE)
@RequiredArgsConstructor
public class PostController {
    private final PostService postService;

    @PostMapping(value = "/post")
    public ApiResponseDto<String> createPost(@RequestBody @Valid PostCreateDto dto) {
        postService.createPost(dto);
        return ApiResponseDto.defaultOk();
    }

    @PostMapping(value = "/post/comment")
    public ApiResponseDto<String> addComment(@RequestBody @Valid PostCommentCreateDto dto) {
        postService.addPostComment(dto);
        return ApiResponseDto.defaultOk();
    }
}
```

Gateway 프로젝트 설정 추가

```yaml
spring:
  cloud:
    gateway:
      mvc:
        routes: # 라우팅 설정
        - id: backend-user
          predicates: # 라우팅 조건
            - Path=/api/user/**
          uri: lb://backend-user
          filters:
            - AddAuthenticationHeader
        - id: backend-alim
          predicates: # 라우팅 조건
            - Path=/api/alim/**
          uri: lb://backend-alim
          filters:
            - AddAuthenticationHeader
        - id: backend-post
          predicates: # 라우팅 조건
            - Path=/api/post/**
          uri: lb://backend-post
          filters:
            - AddAuthenticationHeader
```

application-local.yml