[백엔드 과제수행 easyboard 가이드]

이 프로젝트는 2개의 엔티티와 관련이 있으며 Board는 프로그램이 자동으로 생성합니다(1개의 board 생성). Posting entity 로직과 관련된 작업을 하면 됩니다.

**Entities:**

1. Board
2. Posting

1. PostingLogic

a. registerPosting

1) 파리미터로 전달받은 Cdo 객체를 통해 Entity 객체를 생성하여 저장하세요.

2) 단, 이미 존재하는 엔티티라면 적절한 Exception을 발생시키세요.

3) 저장이 완료되면 저장한 엔티티의 id를 리턴하세요.

b. registerPostings

1) 파라미터로 전달받은 Cdo 객체의 리스트를 통해 Entity를 저장하세요.

c. findPosting

1) 파라미터로 전달받은 Entity의 id를 통해 해당 데이터를 조회 후 리턴하세요.

2) 만약에 해당 쿼리 조건에 부합하는 쿼리결과가 없다면 적절한 Exception을 발생시키세요.

d. findAllPosting

1) 파리미터로 전달받은 Offset 객체를 통해 페이징이 포함된 쿼리 결과 값을 리턴하세요.

e. modifyPosting

1) 파라미터로 전달받은 id를 통해 수정 대상 데이터를 식별하고, NameValues를 통해

수정되어야 하는 값을 수정 후 저장하세요.

f. removePosting

1) 파라미터로 전달받은 id를 통해 해당 데이터를 삭제하세요.

e. exists

1) 파라미터로 전달받은 id를 통해 해당 데이터의 존재 여부를 확인하세요.

2. PostingMariaStore

a. PostingStore 인터페이스의 구현 클래스입니다.

b. Repository 컴포넌트로써 빈으로 생성되어야하는 클래스입니다.

c. PostingMariaRespository를 주입받아 DB 커넥션을 위한 행위를 수행하세요. (JPA)

3. PostingMariaRepostory

a. PostingMariaStore를 구현하면서 추가적으로 필요한 쿼리에 대해 선언하세요.(JPA)

4. PostingResource

a. PostingFacade 인터페이스의 구현 클래스입니다.

b. RestController 컴포넌트로써 빈으로 생성되어야하는 클래스입니다.

c. PostingLogic을 주입받아 필요한 행위를 위임하세요.

d. url 규칙은 주석에 명시된 것을 기준으로 구현하세요.

5. PostingQueryResource

a. PostingQueryFacade 인터페이스의 구현 클래스입니다.

b. RestController 컴포넌트로써 빈으로 생성되어야하는 클래스입니다.

c. PostingStore와 EntityManager를 주입받아 필요한 행위를 로직과 DynamicQuery 객체에 적절히 위임하세요.

d. url 규칙은 주석에 명시된 것을 기준으로 구현하세요.

6. PostingSecureResource

a. PostingResource에 구현된 API 중 프론트를 구현하면서 사용되는 api를 Copy&Paste하세요.

b. 프론트에서 바라보는 url은 ‘secure’ 라는 url 규칙을 가지므로 이와 동일하게

기존의 url 중 aggregate 키워드는 secure로 대치하세요

* 프론트에서 실제로 사용하게 되는 백엔드의 api 입니다.

7. PostingQuerySecureResource

a. PostingQueryResource에 구현된 API 중 프론트를 구현하면서 사용되는 api를 Copy&Paste하세요.

b. 프론트에서 바라보는 url은 ‘secure’ 라는 url 규칙을 가지므로 이와 동일하게

기존의 url 중 aggregate 키워드는 secure로 대치하세요

* 프론트에서 실제로 사용하게 되는 백엔드의 api 입니다.

8. PostingFlowSecureResource

a. PostingFlowResource는 easyboard-flow 모듈에 이미 구현되어 있습니다.

6, 7번과 동일한 방식으로 구현하세요.