Đề Kiểm Tra Kiến Thức Kết Nối CSDL-JPA

1. Thuộc tính name trong annotation @Entity khác với thuộc tính name trong @Table như thế nào? Hãy giải thích rõ cần thì minh hoạ

2. Để debug câu lệnh SQL mà Hibernate sẽ sinh ra trong quá trình thực thi, cần phải bổ sung lệnh nào vào file application.properties?

3. Khi sử dụng H2, làm thế nào để xem được cơ sở dữ liệu và viết câu truy vấn?

4. Khi viết mô tả một model, những thuộc tính chúng ta không muốn lưu xuống CSDL thì cần đánh dấu bằng annotation nào?

5. Annotation @Column dùng để bổ sung tính chất cho cột ứng với một thuộc tính. Tham số nào trong @Column sẽ đổi lại tên cột nếu muốn khác với tên thuộc tính, tham số nào chỉ định yêu cầu duy nhất, không được trùng lặp dữ liệu, tham số nào buộc trường không được null?

6. Có 2 sự kiện mà JPA có thể bắt được, viết logic bổ sung

    - Ngay trước khi đối tượng Entity lưu xuống CSDL (ngay trước lệnh INSERT)
    - Ngay trước khi đối tượng Entity cập nhật xuống CSDL (ngay trước lệnh UPDATE)

    Vậy 2 annotation này là gì

7. Tổ hợp các trường thông tin địa chỉ: country, city, county, addressline thường luôn đi cùng nhau và sử dụng lại trong các Entity khác nhau. Nhóm 2 annotation nào dùng để tái sử dụng, nhúng một Entity vào một Entity khác?

8. JpaRepository là một interface có sẵn trong thư viện JPA, nó cũng cấp các mẫu hàm thuận tiện cho thao tác dữ liệu. Cụ thể JpaRepository kế thừa từ interface nào?
9. Hãy viết khai báo một interface repository thao tác với một Entity tên là Post, kiểu dữ liệu trường Identity là long, tuân thủ interface JpaRepository.
10. Khi đã chọn một cột là Identity dùng @Id để đánh dấu, thì có cần phải dùng xác định unique dùng annotation @Column(unique=true) không?

11. Khác biệt giữa @Id với @NaturalId là gì?
12. Có những cột không phải primary key (@Id) hay @NaturalId, dữ liệu có thể trùng lặp (unique không đảm bảo true), nhưng cần đánh chỉ mục (index) để tìm kiếm nhanh hơn vậy phải dùng annotation gì? Hãy viết 1 ví dụ sử dụng annotation đó với index cho 1 column và 1 ví dụ với index cho tổ hợp nhiều column. Tham khảo tại (https://www.baeldung.com/jpa-indexes)
13. Annotation @GeneratedValue dùng để chọn cách tự sinh unique id cho primary key phải là trường kiểu int hoặc long. Nếu trường primary key có kiểu là String, chúng ta không thể dùng @GeneratedValue vậy hãy thử liệt kê các cách đảm bảo sinh ra chuỗi có tính duy nhất?
14. Giả sử có 1 class Employee với các fields sau {id, emailAddress, firstName, lastName}. Hãy viết các method trong interface EmployeeRespository để :
    - Tìm tất cả các Employee theo emailAddress và lastName
    - Tìm tất cả các Employee khác nhau theo firstName hoặc lastName
    - Tìm tất cả các Employee theo lastName và sắp xếp thứ tự theo firstName tăng dần
    - Tìm tất cả các Employee theo fistName không phân biệt hoa thường
15. Hãy nêu cách sử dụng của @NamedQuery và @Query. Cho ví dụ
16. Làm thế nào để có thể viết custom method implemetations cho Jpa Repository. Nêu ví dụ
17. Hãy nêu 1 ví dụ sử dụng sorting và paging khi query đối tượng Employee ở trên
18. Có 3 Entity Product.java và Category.java và Tag.java
    - Hãy bổ sung định nghĩa quan hệ One to Many giữa bảng Category (One) -- Product (Many). Chú ý khi một Category xoá, thì không được phép xoá Product, mà chỉ set thuộc tính Category của Product là null.
    - Hãy bổ sung định nghĩa quan hệ Many to Many giữa bảng Tag(Many) -- Product(Many).

```
@Entity(name="product")
                          @Table(name="product")
                          @Data
                          public class Product {
                            @Id
                            @GeneratedValue(strategy = GenerationType.AUTO)
                            private long id;

                            private String name;
                          }
```

```
@Entity(name="category")
                           @Table(name="category")
                           @Data
                           public class Category {
                             @Id
                             @GeneratedValue(strategy = GenerationType.AUTO)
                             private long id;

                             private String name;
                           }
```

```
@Entity(name="tag")
                      @Table(name="tag")
                      @Data
                      public class Tag {
                        @Id
                        @GeneratedValue(strategy = GenerationType.AUTO)
                        private long id;

                        private String name;
```

19. Có 2 Entity Student.java và Course.java

```
@Entity(name="student")
                          @Table(name="student")
                          @Data
                          public class Student {
                            @Id
                            private long id;

                            private String name;
                          }
```

```
@Entity(name="course")
                         @Table(name="course")
                         @Data
                         public class Course {
                           @Id
                           private long id;
```

Hãy mô tả quan hệ Many to Many. Một Student có thể học nhiều Course. Một Course có nhiều Student tham gia. Bảng trung gian giữa Student và Course cần có một trường là điểm score kiểu int giá trị gán vào từ 0 đến 10. Cần validate ở phương thức setter

Dữ liệu mẫu để kiểm thử:
student.sql

INSERT INTO student (id, name) VALUES (1, 'bob');
INSERT INTO student (id, name) VALUES (2, 'alice');
INSERT INTO student (id, name) VALUES (3, 'tom');
INSERT INTO student (id, name) VALUES (4, 'jane');
INSERT INTO student (id, name) VALUES (5, 'van');
INSERT INTO student (id, name) VALUES (6, 'long');

course.sql

```
INSERT INTO course (id, name) VALUES (1, 'math');
INSERT INTO course (id, name) VALUES (2, 'music');
INSERT INTO course (id, name) VALUES (3, 'history');
```

Nội dung môn học và điểm

bob học {math: 7, music: 5, history: 8}

alice học {math: 8, music: 2, history: 9}

tom học {math: 4, history: 10}

jane học {music: 9, history: 8}

van học {math: 9, music: 7, history: 6}

long học {math: 10, music: 3}

20. Với kết quả của câu 19: Hãy lập trình JPARepository và viết JUnit5 để tính
    - Trả về liệt kê sinh viên tham gia từng môn học Map<String, List<Student>>: key là tên môn học, value là danh sách sinh viên đăng ký
    - Viết Native Query để tính điểm trung bình một môn bất kỳ
    - Liệt kê danh sách sinh viên học math nhưng không học music