Day-22 Interview Questions

1. What is the purpose of the @Embeddable annotation in Hibernate?

The @Embeddable annotation in Hibernate is used to declare a class as embeddable, which means it can be embedded within an entity. Embeddable objects are used to represent components of an entity, and they don't have a separate database identity.

2. How does Hibernate handle the mapping of @Embeddable objects to database tables?

When an @Embeddable object is embedded within an entity, Hibernate maps its attributes to columns within the same table as the parent entity. The @Embeddable object doesn't have its own database table.

3. Can you provide an example of when you might use @Embeddable in Hibernate?

An example could be using @Embeddable to represent an address within a User entity. The @Embeddable address object can contain attributes like street, city, and postal code, which are stored as columns in the User table.

4. What is the difference between @Embedded and @Embeddable in Hibernate?

@Embeddable is used to declare a class as embeddable, whereas @Embedded is used within an entity class to indicate that an embeddable object should be embedded within that entity.

5. Explain the difference between one-to-one, one-to-many, and many-to-many associations in Hibernate.

One-to-one represents a single related entity, one-to-many represents one entity related to multiple others, and many-to-many represents multiple entities related to multiple others.

6. What are the key differences between a unidirectional and bidirectional association in Hibernate?

In a unidirectional association, one entity knows about the other, while in a bidirectional association, both entities are aware of each other. Bidirectional associations often provide more efficient querying and data consistency.

7. How does Hibernate handle cascading operations in associations, and why is it important?

Cascading operations allow changes in one side of an association to propagate to the other side. For example, if you save a parent entity with a list of child entities, you can set up cascading to automatically save the children. It helps maintain data integrity and simplifies the code.

8. What is the "N+1 select problem," and how can you address it when working with Hibernate associations?

The "N+1 select problem" is a performance issue where, in a one-to-many association, Hibernate fetches data with one query for the parent entity and then N separate queries for each child entity. It can be addressed using techniques like eager loading, batch fetching, or join fetching.

9. In the context of Hibernate associations, what is an "orphan removal" strategy?

Orphan removal is a strategy that, when enabled, automatically deletes child entities that are no longer associated with a parent entity. It helps ensure that the database remains in sync with the application's data model.