Day-21 Interview Questions

1. What is Hibernate?

Hibernate is an open-source, Object-Relational Mapping (ORM) framework that simplifies the interaction between Java applications and relational databases. It maps Java objects to database tables and provides a way to query and manipulate data using Java objects.

2. Explain the different states of an object in Hibernate?

In Hibernate, an object can be in one of three states:

Transient: An object is in the transient state if it's just created and not associated with any Hibernate Session.

Persistent: An object is in the persistent state if it's associated with a Hibernate Session and any changes made to it are tracked.

Detached: An object is in the detached state if it was associated with a Session but the Session is closed. It is no longer managed by Hibernate.

3. What is the architecture of Hibernate?

Hibernate follows a layered architecture:

Application Layer: The top layer where your application interacts with Hibernate through SessionFactory and Session objects.

Persistence Layer: This layer comprises Hibernate's core components, like SessionFactory, Session, and Transaction. It handles database connectivity and interactions.

Database Layer: At the bottom is the actual relational database where data is stored.

4. Explain the role of a Session Factory in Hibernate?

The Session Factory is a crucial component in Hibernate. It is a factory class that produces Session objects. It is thread-safe and should be created only once during the

application's lifecycle. The Session Factory is responsible for initializing Hibernate, managing database connections, and caching metadata.

5. What is a Hibernate Session, and why is it used?

A Hibernate Session is a runtime interface between the Java application and the database. It acts as a factory for creating and managing persistent objects, as well as providing methods for querying the database. A Session is short-lived and is typically created at the start of a unit of work and closed at the end.

6.Explain how Hibernate manages transactions?

Hibernate manages transactions through the following ways:

Programmatic Transactions: You can begin, commit, and rollback transactions manually using the beginTransaction(), commit(), and rollback() methods of the Transaction interface.
Declarative Transactions: You can use annotations or XML configuration to declare transaction boundaries, and Hibernate will manage transactions automatically.
Container-Managed Transactions: If you're using an application server, you can delegate transaction management to the container.

7. What is the difference between get() and load() methods in Hibernate?

get() and load() are both used to load objects from the database, but the main difference is how they handle non-existent objects. get() returns null if the object doesn't exist, whereas load() throws an exception (ObjectNotFoundException) when the object is not found.

8. What is the role of a Hibernate Transaction object?

The Transaction object in Hibernate is responsible for managing the transaction lifecycle. It provides methods to begin, commit, and roll back transactions, ensuring the ACID properties of transactions.

9.What is the purpose of the session.beginTransaction() method in Hibernate?

The session.beginTransaction() method initiates a new transaction for the current Hibernate session. It marks the beginning of a unit of work and is typically followed by

operations that modify the database. The transaction can be committed or rolled back using the commit() or rollback() methods.

10. What is the purpose of the Session.clear() method in Hibernate?

The Session.clear() method detaches all persistent objects from the session, effectively clearing the session's first-level cache. This can be useful in long-running transactions to release memory and prevent objects from becoming stale.

11.How do you handle exceptions and roll back a transaction in Hibernate?

You can use a try-catch block to catch exceptions and then call rollback() on the transaction object to roll back the transaction. This ensures that any changes made during the transaction are not persisted to the database.