Hibernate Most Asked Interview Questions

Q1. What is Hibernate?

Hibernate is an open-source, lightweight, ORM (Object-Relational Mapping) tool in Java which is used to map Java classes to database tables and to convert Java data types to SQL data types.

Q2. What are the core components of Hibernate?

Core components of Hibernate include SessionFactory, Session, Transaction, ConnectionProvider, and TransactionFactory. These components are fundamental in performing database operations through Hibernate framework.

Q3. Explain the role of the SessionFactory in Hibernate.

SessionFactory is a factory class used to create Session objects. It is a heavyweight object meant to be created once per datasource or per database. It is used to open new sessions for interacting with the database.

Q4. What is a Session in Hibernate?

A Session in Hibernate is a single-threaded, short-lived object representing a conversation between the application and the database. It acts as a staging area for changes to be persisted in the database.

Q5. How does Hibernate manage transactions?

Hibernate manages transactions via its Transaction interface. Transactions in Hibernate are handled through a combination of the Java Transaction API (JTA) and JDBC. Hibernate integrates with the transaction management mechanism of the underlying platform.

Q6. What is HQL (Hibernate Query Language)?

HQL stands for Hibernate Query Language, a portable, database-independent query language defined by Hibernate. It is object-oriented, understanding notions like inheritance, polymorphism, and association.

Q7. What is the Criteria API in Hibernate?

The Criteria API is a programmable, object-oriented API in Hibernate used to define complex queries against database entities. It is used to build up a criteria query object programmatically where you can apply filtration rules and logical conditions.

Q8. Explain the concept of Object States in Hibernate.

In Hibernate, objects can exist in one of three states: transient (not associated with any session), persistent (associated with a session), and detached (was once associated with a session but then got detached).

Q9. What is the purpose of the Configuration class in Hibernate?

The Configuration class in Hibernate is used to configure settings from hibernate.cfg.xml file. It bootstraps the Hibernate and allows the application to specify properties and mapping documents to be used when creating a SessionFactory.

Q10. Describe the Second Level Cache in Hibernate.

The Second Level Cache in Hibernate is an optional cache that can store data across sessions. It is used to enhance performance by storing entities in cache memory, reducing database access.

Q11. What are the differences between get() and load() methods in Hibernate?

The get() method in Hibernate retrieves the object if it exists in the database; otherwise, it returns null. The load() method also retrieves the object, but if it doesn't exist, it throws an ObjectNotFoundException. load() can use a proxy to fetch the data lazily.

Q12. How does Hibernate ensure data integrity?

Hibernate ensures data integrity by managing database transactions, providing isolation levels, and supporting concurrency strategies. It also integrates with database constraints and can enforce application-level integrity using validators.

Q13. What is the N+1 SELECT problem in Hibernate? How can it be prevented?

The N+1 SELECT problem in Hibernate occurs when an application makes one query to retrieve N parent records and then makes N additional queries to retrieve related child objects. It can be prevented using strategies like join fetching, batch fetching, or subselect fetching to minimize the number of queries executed.

Q14. Explain the role of the @Entity annotation in Hibernate.

The @Entity annotation in Hibernate is used to mark a class as an entity, which means it is a mapped object and its instance can be persisted to the database.

Q15. What is cascading in Hibernate?

Cascading in Hibernate is the ability to propagate the operations from a parent entity to its associated child entities. It is used to manage the state transitions of associated objects automatically. CascadeType can be used to specify which operations are cascaded.

Q16. What is a Composite Key in Hibernate?

A Composite Key in Hibernate is a primary key made up of multiple columns. In Hibernate, a composite key can be represented using a separate class annotated with @Embeddable or @EmbeddedId to represent this composite key.

Q17. How does Hibernate handle SQL Injection?

Hibernate handles SQL Injection by using prepared statements that automatically escape SQL syntax. Additionally, using HQL or Criteria API also protects against SQL injection as they translate a query from HQL into SQL in a way that uses parameterized queries.

Q18. What is Lazy Loading in Hibernate?

Lazy Loading in Hibernate is a concept where an entity or collection of entities is not loaded until it is accessed for the first time. This is a performance optimization technique to defer the loading of objects until they are needed.

Q19. How can you achieve concurrency in Hibernate?

Concurrency in Hibernate can be achieved using versioning and locking mechanisms. Hibernate supports optimistic and pessimistic locking strategies to handle concurrent modifications of data effectively.

Q20. What is an optimistic locking in Hibernate?

Optimistic locking in Hibernate is a technique to ensure that a record is not updated by more than one transaction at the same time by using a version field in the database table. It checks the version of a record at the time of fetching and before committing an update to ensure consistency.

Q21. You have noticed that your Hibernate application is running slowly when fetching data from a database with many relationships. What strategy could you use to improve performance?

To optimize query performance in Hibernate, I would consider using lazy loading for entity relationships. This means Hibernate will only fetch related entities when they are explicitly accessed, not at the time of fetching the parent entity. Additionally, I might use batch fetching and adjust the fetch sizes in the configuration to reduce the number of database queries.

Q22. How do you handle a Hibernate session in a web application to ensure that it is properly closed, avoiding memory leaks?

In our web application, we manage Hibernate sessions by binding a session to the current thread using the CurrentSessionContext interface. We typically configure session opening and closing in a servlet filter or interceptors, ensuring that each request opens a session and ends by closing the session, thus preventing memory leaks.

Q23. During a transaction, an error occurs after several database operations have been successfully executed. How does Hibernate ensure data integrity in this situation?

Hibernate ensures data integrity by using transactions. If an error occurs during the transaction, hibernate rolls back all operations to the state before the transaction began, using either database transactions or the Java Transaction API (JTA). This rollback mechanism prevents partial data modifications that could lead to data inconsistency.

Q24. You need to add auditing features to track changes in entity data. What Hibernate feature would you use to achieve this?

To implement auditing in Hibernate, I would use Hibernate Envers. It's a Hibernate module that allows for versioning of entity classes. By simply annotating our entity classes with @Audited, we can keep track of changes to their state, automatically storing revisions in separate tables.

Q25. You are working with a legacy database where the table and column names do not follow your standard naming conventions. How can you map these tables without modifying the existing database schema?

In Hibernate, I handle legacy databases by customizing the ORM mapping. I use the @Table and @Column annotations to map entity classes to the specific table names and column names defined in the legacy database. This allows us to map the entities accurately to the database schema without any changes to the database itself.