**HIBERNATE**

**Q1. What is Hibernate?**

Hibernate -is one of the most popular Java frameworks that simplify the development of Java application to interact with the database. It is an Object-relational mapping (ORM) tool. Hibernate also provides a reference implementation of Java API.

It is referred as a framework which comes with an abstraction layer and also handles the implementations internally. The implementations include tasks like writing a query for CRUD operations or establishing a connection with the databases, etc.

Hibernate develops persistence logic, which stores and processes the data for longer use. It is a lightweight tool and most importantly open-sourced which gives it an edge over other frameworks.

**Q2. What are the major advantages of Hibernate Framework?**

1. It is open-sourced and lightweight.
2. Performance of Hibernate is very fast.
3. Helps in generating database independant queries.
4. Provides facilities to automatically create a table.
5. It provides query statistics and database status.

**Q3. What are the advantages of using Hibernate over JDBC?**

Major advantages of using Hibernate over JDBC are:

Hibernate eliminates a lot of boiler-plate code that comes with JDBC API, the code looks cleaner and readable.

This Java framework supports inheritance, associations, and collections. These features are actually not present in JDBC.

HQL (Hibernate Query Language) is more object-oriented and close to Java. But for JDBC, you need to write native SQL queries.

Hibernate implicitly provides transaction management whereas, in JDBC API, you need to write code for transaction management using commit and rollback.

JDBC throws SQLException that is a checked exception, so you have to write a lot of try-catch block code. Hibernate wraps JDBC exceptions and throw JDBCException or HibernateException which are the unchecked exceptions, so you don’t have to write code to handle it has built-in transaction management which helps in removing the usage of try-catch blocks.

**Q4. What is Session in Hibernate and how to get it?**

Hibernate Session is the interface between Java application layer and Hibernate. It is used to get a physical connection with the database. The Session object created is lightweight and designed to be instantiated each time an interaction is needed with the database. This Session provides methods to create, read, update and delete operations for a constant object. To get the Session, you can execute HQL queries, SQL native queries using the Session object.

**Q5. What is Hibernate SessionFactory?**

SessionFactory is the factory class that is used to get the Session objects. The SessionFactory is a heavyweight object so usually, it is created during application startup and kept for later use. This SessionFactory is a thread-safe object which is used by all the threads of an application. If you are using multiple databases then you would have to create multiple SessionFactory objects.

**Q6. What is the difference between openSession and getCurrentSession?**

This getCurrentSession() method returns the session bound to the context and for this to work, you need to configure it in Hibernate configuration file. Since this session object belongs to the context of Hibernate, it is okay if you don’t close it. Once the SessionFactory is closed, this session object gets closed.

**Q7. What are Inheritance Mapping Strategies available in Hibernate?**

We can map the inheritance hierarchy classes with the table of the database. There are three inheritance mapping strategies defined in the hibernate:

**Table Per Hierarchy**

In table per hierarchy mapping, single table is required to map the whole hierarchy, an extra column (known as discriminator column) is added to identify the class. But nullable values are stored in the table .

**Table Per Concrete class**

In case of table per concrete class, tables are created as per class. But duplicate column is added in subclass tables.

**Table Per Subclass**

In this strategy, tables are created as per class but related by foreign key. So there are no duplicate columns.

**Q8. Difference between the first and second level cache in Hibernate?**

The first-level cache is maintained at Session level while the second level cache is maintained at a SessionFactory level and is shared by all sessions.

| **Sr. No.** | **Key** | **First level cache** | **Second level cache** |
| --- | --- | --- | --- |
| 1 | Basic | First level cache is a session level cache and it is always associated with session level object | Second level cache is session factory level cache and it is available across all sessions |
| 2 | Enabled | It is enabled by default. | It is not enabled by default. |
| 3 | Availability | It is available for a session | It is available across all session. |
| 4 | Configuration | No Extra configuration required | We have to decide  which concurrency strategy to use and also need to configure cache expiration and physical cache attributes. |

**10. What is the difference between a transient, persistent, and detached object in Hibernate?**

In Hibernate, Objects can remain in three states transient, persistent, or detached. An object which is associated with the Hibernate session is called a persistent object.

Any change in this object will reflect in the database based on your flush strategy i.e. automatic flush whenever any property of object changes or explicit flushing by calling Session.flush() method.

On the other hand, if an object which is earlier associated with Session, but currently not associated with it is called a detached object.

**Q11. Explain the Key components of Hibernate?**

The Key components of Hibernate are as described.

1. Session: Session is used to get a physical network with a database.
2. Query: It uses SQL and HQL string to retrieve the data from the database and create objects.
3. Transaction: Transaction represents the unit of work with a database.
4. Criteria: The primary use of criteria is to create and execute object-oriented queries and retrieve the objects.
5. Configuration: It represents the properties of files required by Hibernate
6. Session Factory: It configures hibernate for the application using the provided configuration file and instantiates the session object.

**Q12. Explain Session?**

The session object is used to connect with a database. It is light-weight and is instantiated every time a database connection is required. Sessions are not thread safe and so should not be kept open for long. These objects should be created as needed and should be destroyed if not in use. Sessions hold a first-level cache of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

The main functionality of Session objects is to provide create, read and delete operations to the instances of the mapped entity classes. The instances may be in one of the following states at any given point.

1. Transient: When an instance of a persistent class is not associated with a session, has no database representation and has no identifier value, such an instance is considered to be in the transient state
2. Persistent: An instance becomes persistent when it is associated with a session, has database representation and has an identifier value
3. Detached: An instance is detached, once the Hibernate session is closed.

A session instance becomes serializable if its persistent class or classes is/are serialised. If a session throws an error, the transaction has to be rolled back and the session instance has to be deleted.

**Q13. What are the core interfaces of Hibernate?**

The core interfaces of Hibernate framework are used to configure the application, store and retrieve objects and to control transactions. The interfaces are:

1. Configuration: The Configuration objects are used to configure the application by setting the configuration file and mapping documents location, retrieving the Hibernate - specific properties etc It also helps create the SessionFactory.
2. SessionFactory: The SessionFactory provides obtains Session instances to the application. SessionFactory instances are not lightweight and usually one instance is created for the whole application. If the application accesses multiple databases, it needs one per database. SessionFactory is thread-safe and can holdsecond-level cache of data that is reusable between transactions at a process, or cluster, level.
3. Session: The Session is a persistence manager that manages operation like storing and retrieving objects. Instances of Session are inexpensive to create and destroy. Session objects are not thread safe and have to be destroyed once their usefulness is over. Session holds a mandatory first-level cache of persistent objects that are used when navigating the object graph or looking up objects by identifier.
4. Query: This interface is used to query the database and control how the query execution takes place. Queries are written using HQL or in the native SQL language of the database being used. A Query instance allows for the binding of query parameters, limiting the number of results returned by the query, and executing the query. A Query instance cannot be used outside the Session in which it has been created
5. Criteria: This interface is used to create and execute queries which are criteria based. Such criteria-based queries are called object-oriented criteria queries.
6. Transaction: A transaction represents a unit of work. When an application performs some operations on a database, it is called a transaction. We can have multiple operations within a single transaction and a single commit to commit all the operations at once. If a transaction fails or if an operation in a transaction fails, the transaction can be rolled back. This interface is an optional interface which the application may use or the application may choose to handle its DB operations and transactions through its own code.

**Q15. How many types of association mapping are possible in hibernate?**

There can be 4 types of association mapping in hibernate.

1. One to One: A one-to-one association is similar to many-to-one association with a difference that the column will be set as unique. For example, an address object can be associated with a single employee object. <one-to-one> element is used to define one-to-one association. The name attribute is set to the defined variable in the parent class. The column attribute is used to set the column name in the parent table which is set to unique so that only one object can be associated with another object.
2. One to Many: In One-to-Many mapping association, an object can be associated with multiple objects. For example, Employee object relates to many Certificate objects. A One-to-Many mapping can be implemented using a Set java collection that does not contain any duplicate element. <one-to-many> element of set element indicates that one object relates to many other objects.
3. Many to One: A many-to-one association is the most common kind of association where an Object can be associated with multiple objects. For example, a same address object can be associated with multiple employee objects. <many-to-one> element is used to define many-to-one association. The name attribute is set to the defined variable in the parent class. The column attribute is used to set the column name in the parent table.
4. Many to Many: A Many-to-Many mapping can be implemented using a Set java collection that does not contain any duplicate element. <many-to-many> element indicates that one object relates to many other objects and column attributes are used to link intermediate column.

**Q16. Explain Hibernate configuration file and Hibernate mapping file?**

Hibernate configuration file: This file contains a set of configuration settings related to the database and other related parameters which is required by Hibernate for connecting to the database. This file is placed in the root directory of the application’s classpath. Some properties that are configured in this file are:

1. hibernate.connection.driver\_class: which is the JDBC driver class for database connection
2. hibernate.connection.url: which is the url to the database instance
3. hibernate.connection.username: is the username to the database instance
4. hibernate.connection.password: is the password to the database instance
5. hibernate.connection.autocommit: is to allow autocommit mode to the JDBC connection

Hibernate Mapping file: This file contains the mapping of a POJO class to the table. Giving the POJO class the same name as the table and having class properties corresponding to the column names helps in avoiding confusions while mapping the class properties to the table columns.An object normally contains 3 properties - identity, state and behavior. The identity and behavior of the object has to be avoided when storing the object state (value) in the database. This is taken care of by mapping. A Hibernate application can have multiple mapping files for mapping the classes to the tables.Mapping can be done using XML or Annotations.

**Q18. How to achieve mapping in Hibernate?**

Hibernate offers various features with association mapping being one of them. This framework supports the same associations as the relational database model. These associations can be mapped as a uni-directional or bi-directional association. These associations are as follows:

One-to-One associations

Many-to-One associations

Many-to-Many associations

**Q19. What is the One-to-One association in Hibernate framework?**

In One-to-one mapping, the developer is required to model the system for the entity for which they want to navigate the relationship in their query or domain model. The developer needs an entity attribute that represents the association. Hence, it must be annotated with a @OneToOne annotation.

**Q20. What is the Many-to-Many association in hibernate framework?**

A @ManytoMany annotation and an entity attribute are required in the Many-to-Many mapping. This association can either be uni-directional or bi-directional. Under unidirectional attributes, the attributed model the association and the developers use the association to navigate it in their domain model or JPQL queries. The annotation plays the role of telling the framework to map a Many-to-Many association. Under the bidirectional association, the Many-to-Many mapping helps the developer in navigating the association in both directions.

**Q21. Explain Hibernate Configuration File?**

Hibernate Configuration File is also known as hibernate.cfg.xml. This is one of the most important and commonly used configuration files in Hibernate framework. This configuration file is stored under the src/main/resource folder, by default. The hibernate.cfg.xml contains configuration related to the database and configuration related to the session. Hibernate framework provides the configuration in 2 files, either in an XML file such as hibernate.cfg.xml or a property file such as hibernate.properties.

The hibernate configuration file is used to ascertain the following information:

It is used to define database connection details such as driver class, URL, username and password.

Each database used in the application has one configuration. To connect with two databases,2 configuration files with different names must be created.

The hibernate framework properties are: dialect, show\_sql, second\_level\_cache, and mapping file names.

**Q22. How does one create an immutable class in the hibernate framework?**

The immutable class in the hibernate framework is used as follows:

If the developer uses the XML form of the configuration, a class can be made immutable with the help of the markingmutable=false.

The default value of the element is true. Hence, it indicates that the class is not created by default.

The developer can also use annotations to make an immutable class in hibernate. It is created by using the annotation @Immutable.

**Q23. Explain criteria API in Hibernate.**

In hibernate framework, the criteria API helps developers build dynamic criteria queries on a persistence database. Criteria API is more powerful and flexible than Hibernate Query Language queries to create dynamic queries.

The criteria API helps in programmatically developing criteria query objects. Here, the interface org.hibernate.Criteria are used for all purposes. Also, the createCritera() method in the session interface is executed. It takes the persistent object’s class or its entity name as the parameters and returns the persistence object when the criteria are executed.

The API also makes it easy to inject restrictions to selectively retrieve data from the database. This can be done by using the add() method. The org.hibernate.criterion.Criterion object is used in representing individual restriction.