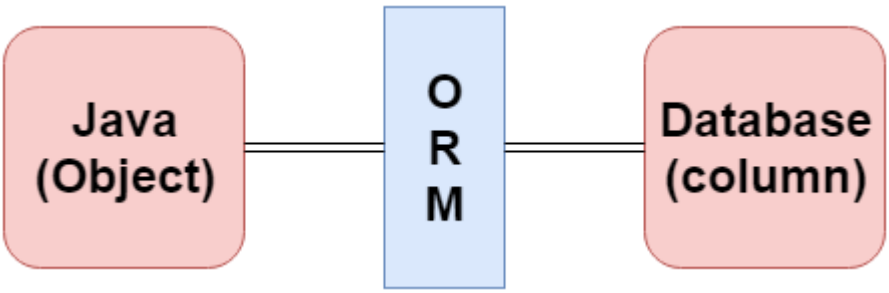
Q1. What is ORM in Hibernate?

Answer: Object Relational Mapping (ORM) is a functionality which is used to develop and maintain a relationship between an object and relational database by mapping an object state to database column. It is capable to handle various database operations easily such as inserting, updating, deleting etc.



ORM Frameworks

Following are the various frameworks that function on ORM mechanism: -

- Hibernate
- TopLink
- ORMLite
- iBATIS
- JPOX

Mapping Directions

Mapping Directions are divided into two parts: -

- Unidirectional relationship - In this relationship, only one entity can refer the properties to another. It contains only one owing side that specifies how an update can be made in the database.
- Bidirectional relationship - This relationship contains an owning side as well as an inverse side. So here every entity has a relationship field or refer the property to other entity.

Types of Mapping

Following are the various ORM mappings: -

- One-to-one - This association is represented by @OneToOne annotation. Here, instance of each entity is related to a single instance of another entity.
- One-to-many - This association is represented by @OneToMany annotation. In this relationship, an instance of one entity can be related to more than one instance of another entity.
- Many-to-one - This mapping is defined by @ManyToOne annotation. In this relationship, multiple instances of an entity can be related to single instance of another entity.
- Many-to-many - This association is represented by @ManyToMany annotation. Here, multiple instances of an entity can be related to multiple instances of another entity. In this mapping, any side can be the owing side.

Q2. What are the advantages of Hibernate over JDBC?

Answer:

1. Database Dependency

The Hibernate is not dependent on the database whereas in the case of JDBC the queries that need to be written are dependent on the database.So, if we are using some database like MySQL or PostgreSQL in beginning but in the later stage, we have the liability of switching to another database like oracle.

It is so because hibernate creates XML configuration files therefore only these files are changed instead of the complete query.

Example:

For SQL:

SELECT TOP 10 name_of_column FROM name_of_table ORDER BY name_of_column ASC

For PostgreSQL:

SELECT name_of_column FROM name_of_table ORDER BY name_of_column ASC LIMIT 10

For Hibernate:

Session.CreateQuery("SELECT t. name_of_column FROM name_of_table t ORDER BY t.name_of_column ASC").SetMaxResults(10).List()

There is no change required in the query for hibernate.

2. Optimized Code

Hibernate creates more shorter and optimized code. For example, by creating the connection while using Hibernate the developer should not need to write the complex queries because HCL is there to ease the written query whereas in JDBC it is the complete responsibility of the developer to program the queries.

3. Caching

Hibernate comes up with the caching mechanism. When a query is running multiple times then it can store the value instead of checking it repeatedly from the database. It is responsible to increase the performance of your system whereas in the case of JDBC there isn't any caching mechanism available.

Hibernate comes up with two types of caching levels which are first level and second level.

4. Lazy Loading

Lazy loading of data can be achieved with Hibernate. Lazy loading is a kind of data fetching mechanism and it helps in increasing performance. For example, if we have multiple child classes for parent classes but we need only some specific child classes to get loaded. Therefore, through lazy loading, we can select some of the specific classes for execution.

5. Associations

Associate mapping can be done while using Hibernate. We can use associations like one to one and one to many through annotations. This helps to manage the entity in an easier way.

Q3. What are some of the important interfaces of Hibernate framework?

Answer: Some important interfaces of Hibernate framework include:

SessionFactory: It is a factory for creating Session objects. It is typically created once during application startup and shared by all application threads.

Session: It represents a single-threaded unit of work and provides methods for performing database operations.

Transaction: It represents a database transaction and provides methods for managing transaction boundaries.

Query: It is used to perform queries against the database using either HQL or native SQL.

Criteria: It provides a simplified API for creating queries based on specific criteria.

Configuration: It represents the configuration settings for Hibernate and is used to create a SessionFactory.

Q4. What is a Session in Hibernate?

Answer: In Hibernate, a Session represents a single-threaded unit of work with the database. It is obtained from a SessionFactory and provides methods for performing database operations such as saving, updating, deleting, and querying entities. The Session acts as a context for persistent objects and tracks changes made to them. It also implements a first-level cache, known as the "first level cache" or "session cache," which stores objects retrieved from the database during the session. The Session manages the state of persistent objects, synchronizes changes with the database, and provides transactional support.

Q5. What is a SessionFactory?

Answer: SessionFactory is an important component in Hibernate. It is a factory for creating Session objects. The SessionFactory is typically created once during the application startup and shared by all application threads. It is responsible for bootstrapping Hibernate, reading the configuration settings, and creating connections to the database. It is thread-safe and allows multiple sessions to be created. The SessionFactory caches metadata about the mapping between Java objects and database tables, which improves performance by reducing the overhead of repeated metadata lookups.

Q6. What is HQL?

Answer: HQL (Hibernate Query Language) is a query language provided by Hibernate. It is similar to SQL but uses object-oriented concepts and syntax to perform queries on entities mapped in Hibernate. HQL allows developers to write database-independent queries using entity names, properties, and relationships, rather than dealing with low-level SQL. HQL queries are translated by Hibernate into SQL queries specific to the underlying database. HQL supports various features such as joins, aggregations, projections, and sorting.

Q7. What are Many to Many associations?

Answer: Many-to-Many associations in Hibernate represent a relationship between two entities where each instance of one entity can be associated with multiple instances of the other entity, and vice versa. For example, consider two entities: "Student" and "Course." A many-to-many association between them means a student can enroll in multiple courses, and a course can have multiple students. In Hibernate, this association is typically represented by creating a join table that contains foreign key references to both entities. Hibernate provides mechanisms to handle many-to-many associations, including cascading operations, fetch strategies, and bidirectional associations.

Q8. What is hibernate caching?

Answer: Hibernate caching refers to the mechanism of storing frequently accessed data in memory to improve application performance. Hibernate provides different levels of caching:

First Level Cache: Also known as the session cache, it is enabled by default and operates within the scope of a session. The first level cache stores the objects retrieved or persisted by the session. It helps in reducing the number of database queries by keeping a copy of the loaded objects in memory. The first level cache is associated with the session and is cleared when the session is closed or cleared explicitly.

Second Level Cache: The second level cache is a shared cache that can be used by multiple sessions. It is enabled by configuring a cache provider in Hibernate's configuration file. The second level cache stores objects at the session factory level and can be shared across multiple sessions. It helps in improving performance by reducing the number of database queries for common or repeated data access. The second level cache is more persistent than the first level cache and can survive the lifespan of multiple sessions.

Q9. What is the difference between first level cache and second level cache?

Answer: The difference between the first level cache and the second level cache in Hibernate is as follows:

Scope: The first level cache operates within the scope of a session. It is associated with a specific Session object and is cleared when the session is closed or cleared explicitly. The second level cache, on the other hand, operates at the session factory level and can be shared across multiple sessions. It can persist data across multiple sessions.

Granularity: The first level cache is finer-grained and stores individual objects loaded or persisted within a session. It provides object-level caching. The second level cache is coarser-grained and stores objects at the session factory level. It provides caching at the entity or collection level.

Lifetime: The first level cache is short-lived and exists as long as the associated session is open. It is cleared when the session is closed or cleared explicitly. The second level cache is long-lived and can survive the lifespan of multiple sessions. It can be configured to have a specific expiration policy or eviction strategy.

Q10.What can you tell about Hibernate Configuration File?

Answer: The Hibernate Configuration File, typically named hibernate.cfg.xml, is an XML file that contains the configuration settings for Hibernate. It is used to configure various aspects of Hibernate, such as database connection properties, mapping files, caching options, transaction settings, and more. The configuration file specifies the necessary information for Hibernate to establish a database connection and perform ORM operations. It also provides the mapping between Java classes and database tables, defining how the objects are persisted and retrieved from the database. The Hibernate Configuration File is an essential component for bootstrapping Hibernate and is usually located in the classpath of the application.