Assignment Questions 8

Q1. What is ORM in Hibernate?

ORM stands for **O**bject-**R**elational **M**apping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C#, etc.

Q2. What are the advantages of Hibernate over JDBC?

Major advantages of using Hibernate over JDBC are:

1. Hibernate eliminates a lot of boiler-plate code that comes with *JDBC API*, the code looks cleaner and readable.
2. This Java framework supports *inheritance*, associations, and collections. These features are actually not present in JDBC.
3. HQL (Hibernate Query Language) is more object-oriented and close to Java. But for JDBC, you need to write native SQL queries.
4. Hibernate implicitly provides transaction management whereas, in JDBC API, you need to write code for transaction management using *commit* and *rollback*.
5. 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.

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

Hibernate interfaces are:

- **SessionFactory** (org.hibernate.SessionFactory)
- **Session** (org.hibernate.Session)
- **Transaction** (org.hibernate.Transaction)

Q4. What is a Session in Hibernate?

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 a 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 HQL?

HQL is the acronym of Hibernate Query Language. It is an Object-Oriented Query Language and is independent of the database.

Q7. What are Many to Many associations?

Many-to-Many mapping requires an entity attribute and a *@ManyToMany* annotation. It can either be unidirectional and bidirectional. In **Unidirectional**, the attributes model the association and you can use it to navigate it in your domain model or JPQL queries. The annotation tells Hibernate to map a Many-to-Many association. The **bidirectional** relationship, mapping allows you to navigate the association in both directions.

Q8. What is hibernate caching?

Hibernate caching improves the performance of the application by pooling the object in the cache. It is useful when we have to fetch the same data multiple times.

There are mainly two types of caching:

- First Level Cache, and
- Second Level Cache

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

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.

Q10. What can you tell about Hibernate Configuration File?

The following steps help in configuring Hibernate file:

1. First, identify the POJOs (Plain Old Java Objects) that have a database representation.
2. Identify which properties of POJOs need to be continued.
3. Annotate each of the POJOs in order to map the Java objects to columns in a database table.
4. Create a database schema using the schema export tool which uses an existing database, or you can create your own database schema.
5. Add Hibernate Java libraries to the application's classpath.
6. Create a Hibernate *XML configuration file* that points to the database and the mapped classes.
7. In the Java application, you can create a Hibernate Configuration object that refers to your XML configuration file.
8. Also, build a Hibernate SessionFactory object from the Configuration object.
9. Retrieve the Hibernate Session objects from the SessionFactory and write down the data access logic for your application (create, retrieve, update, and delete).