# **Assignment 8**

Q1. What is ORM in Hibernate?

**ORM** stands for **Object Relation Mapping.** It is a middleware application or tool that sits between the web application and database.

**ORM** is a technique for converting data between Java objects and relational databases (table). In simple words, we can say that the ORM implements responsibility of mapping the object to relational model and vice-versa.

The ORM tool does mapping in such a way that model class becomes a table in the database and each instance becomes a row of the table.

There are many ORM tools available, but the following ORM tools are the most used.

* Hibernate
* TopLink
* EclipseLink
* OpenJPA

Q2. What are the advantages of Hibernate over JDBC?

1. Developer must write code in JDBC to map an object model's data representation to a relational data model and its corresponding database schema.  
  
2. Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this.  
JDBC supports only native Structured Query Language (SQL).  
Hibernate provides Hibernate Query Language (HQL) which is similar to SQL syntax and supports polymorphic queries too. It also supports native SQL statements.  
  
3. The mapping of Java objects with database tables has to be taken care of in JDBC. Hibernate provides transparent persistence and therefore there is no need to map database tables tuples to application objects during interaction with RDBMS.  
  
4. With JDBC, caching needs to be manually maintained. Hibernate cache is set to application workspace. Relational tuples are moved to this cache because of query. It improves performance during multiple writes for the same data.  
  
5. In JDBC there is no check that always every user has updated data. Hibernate enables definition of version type field to application, due to which Hibernate updates version field of database table every time relational tuple is updated in form of Java class object to that table.

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

**1. Session Interface:** The basic interface for all hibernate applications. The instances are light weighted and can be created and destroyed without expensive process.  
  
**2. SessionFactory** **interface :** The delivery of session objects to hibernate applications is done by this interface. For the whole application, there will be generally one SessionFactory and can be shared by all the application threads.  
  
**3. Configuration** **Interface :** Hibernate bootstrap action is configured by this interface. The location specification is specified by specific mapping documents, is done by the instance of this interface.  
  
**4. Transaction** **Interface** : This is an optional interface. This interface is used to abstract the code from a transaction that is implemented such as a JDBC / JTA transaction.  
  
**5. Query and Criteria** **interface :** The queries from the user are allowed by this interface apart from controlling the flow of the query execution.

Q4. What is a Session in Hibernate?

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed. The main function of the Session is to offer, create, read, and delete operations for instances of mapped entity classes.

Instances may exist in one of the following three states at a given point in time −

* **transient** − A new instance of a persistent class, which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.
* **persistent** − You can make a transient instance persistent by associating it with a Session. A persistent instance has a representation in the database, an identifier value and is associated with a Session.
* **detached** − Once we close the Hibernate Session, the persistent instance will become a detached instance.

Q5. What is a SessionFactory?

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

Configuration object is used to create a SessionFactory object which in turn configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all the threads of an application.

The SessionFactory is a heavyweight object; it is usually created during application start up and kept for later use. You would need one SessionFactory object per database using a separate configuration file. So, if you are using multiple databases, then you would have to create multiple SessionFactory objects.

Q6. What is HQL?

Hibernate Query Language (HQL) is an easy to learn and powerful query language designed as an object-oriented extension to SQL that bridges the gap between the object-oriented systems and relational databases. The HQL syntax is very similar to the SQL syntax.

HQL can also be used to retrieve objects from database through O/R mapping by performing the following tasks:

* Apply restrictions to properties of objects
* Arrange the results returned by a query by using the order by clause
* Paginate the results
* Aggregate the records by using group by and having clauses
* Use Joins
* Create user-defined functions
* Execute subqueries

Q7. What are Many to Many associations?

A many-to-many association is made between two entities where one entity can be associated with multiple other instances of the other entity.

Example

**subscription service**, SubscriptionEntity and ReaderEntity can be two types of entities. A given subscription can have multiple readers, whereas a reader can subscribe to multiple subscriptions.

Q8. What is hibernate caching?

Caching is a mechanism to enhance the performance of a system. It is a buffer memory that lies between the application and the database. Cache memory stores recently used data items in order to reduce the number of database hits as much as possible.

## **Cache Provide**r

## EHCache

OSCache

JBoss Cache

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

|  |  |  |  |
| --- | --- | --- | --- |
| **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 must decide which concurrency strategy to use and also need to configure cache expiration and physical cache attributes. |

Q10.What can you tell about the Hibernate Configuration File?

As Hibernate can operate in different environments, it requires a wide range of configuration parameters. These configurations contain the mapping information that provides different functionalities to Java classes.

Generally, we provide database related mappings in the configuration file. Hibernate facilitates to provide the configurations either in an XML file (like hibernate.cfg.xml) or properties file (like hibernate.properties).

An instance of Configuration class allows specifying properties and mappings to applications. This class also builds an immutable **SessionFactory**.

We can acquire the Configuration class instance by instantiating it directly and specifying mappings in the configuration file. Use the addResource() method, if the mapping files are present in the classpath.

Configuration cfg=new Configuration().addResource("employee.hbm.xml")

**Properties File Configuration**

1. hibernate.dialect= org.hibernate.dialect.Oracle9Dialect
2. hibernate.connection.driver\_class= oracle.jdbc.driver.OracleDriver
3. hibernate.connection.url= jdbc:oracle:thin:@localhost:1521:xe
4. hibernate.connection.username= system
5. hibernate.connection.password=jtp
6. hibernate.show\_sql=**true**
7. hibernate.hbm2ddl=update