**Q:1 What is Hibernate?**

Ans: Hibernate is an ORM (Object Relational Mapping) and persistent framework. The framework helps to map plain java object to relational database table using xml configuration file.

The framework helps to perform following things.

• Perform basic CURD operations.  
• Write queries referring to java classes (HQL queries).  
• Facilities to specify metadata.  
• Dirty checking, lazy association fetching.

**Q: 2 Why hibernate and how does it help in the programming?**

Ans: The main advantage of Hibernate (ORM) framework is that it shields developer to write a messy SQL. Apart from that ORM provides following benefits.

• Improve Productivity of the developer by providing high level object oriented API (e.g. API for easily maintaining the connection to data base, mapping java classes to relational database tables), less java code to write, helps to avoid writing SQL query.  
• Improved performance by providing sophisticated caching, lazy loading and eager loading features.  
• Provide portability, the framework helps to generate database specific SQL for you.

**Q:3 How will you configure Hibernate?**

Ans: To configure hibernate, you need hibernate.cfg.xml or hibernate.properties file and \*.hbm.xml files, all these files are used by Configuration class to create sessionFactory, which in turn creates the session instances. Session instances are the primary interface for persistence services.

The hibernate.cfg.xml or hibernate.properties files are used to configure the hibernate service (database connection driver class, database connection URL, connection user name, connection password, dialect, mapping resources etc.).

The \*hbm.xml files are used for mapping persistent objects to relational database.

From Java 5 onwards you can configure and map persistent objects through annotations.

**Q:4 Which settings will be loaded if both hibernate.properties and hibernat.cf.xml files are present in the classpath?**

Ans: If both hibernate.properties and hibernate.cfg.xml files are present in the classpath then hibernate.cfg.xml file will override the settings found in hibernate.properties. So please make sure that your project should include either hibernate.properties or hibernate.cfg.xml file.

**Q:5 What are the Core interfaces of Hibernate framework?**

Ans: There are five core interfaces being used extensively in every Hibernate application. Using these interfaces you can store or retrieve any persistent objects and also control transactions.

- Session interface  
- SessionFactory interface  
- Configuration interface  
- Transaction interface  
- Query and Criteria interfaces

**Q:6 What is the role of Session interface in Hibernate?**

The session is a interface. It is single threaded sort-lived object and represents conversation between Application and the persistent store. It helps to create query objects, to retrieve persistent objects.

You can get the session object from session factory

Session session = sessionFactory.openSession();

Session Interface role:

--Wraps a JDBC connection  
--Factory for Transaction  
--Holds a mandatory (first-level) cache of persistent objects, used when navigating the object graph or looking up objects by identifier

**Q: 7 What is the role of SessionFactory?**

Ans: The application obtains session object from SessionFactory interface. Typically there should be only one sessionFacory for whole application and is loaded during application initialization. The SessionFactory caches generate SQL Statement and other mapping metadata that Hibernate use at runtime. It also hold cached data that has been read in one unit of work and can be reused in a future unit of work.

You can get the instance of SessionFactory by the configuration object as below

SessionFactory sessionFactory = configuration.buildSessionFactory();

**public** **class** HibernateUtil {

**private** **static** **final** SessionFactory *sessionFactory* = *buildSessionFactory*();

**private** **static** SessionFactory buildSessionFactory() {

**try** {

// Create the SessionFactory from hibernate.cfg.xml

**return new Configuration().configure().buildSessionFactory();**

} **catch** (Throwable ex) {

// Make sure you log the exception, as it might be swallowed

System.*err*.println("Initial SessionFactory creation failed." + ex);

**throw** **new** ExceptionInInitializerError(ex);

} }

}

**public** **static** SessionFactory getSessionFactory() { **return** *sessionFactory*;}

**Q:13 What are the Collection types in Hibernate?**

Ans:  
• Bag  
• Set  
• List  
• Array  
• Map

**Q:14 What is HibernateTemplate?**

Ans: The spring framework provides HibernateTemplate (org.springframework.orm.hibernate.HibernateTemplate) which is kind of helper class and provides following benefits.

- HibernateTemplate class simplifies interaction with Hibernate session.  
- Common functions are simplified to single method calls.  
- Sessions are automatically closed.  
- Exception are automatically caught and converted to runtime exceptions.

**Q:15 What is the difference between load() and get() method?**

**load():**  
- Use load() method only when you are sure that object you want to read already exists.   
- If unique Id of an object does not exists in database then load() method will throw an exception.  
- load() method return proxy object default and database won't be hit until the proxy is first invoked.

**get():**

- Use get() method when you are not sure about the object existance in the database.  
- If object does not exists in the database, the get() method will return null.  
- get() method will hit database immediately.

**Q:16 What is lazy fetching in Hibernate?**

Ans: In Hibernate Lazy fetching is associated with child objects loading for its parents. Through Hibernate mapping file (.hbm.xml) you can specified the selection of loading child objects. By default Hibernate does not load child objects. Lazy=true means not to load the child objects.

#### What is lazy initialization in hibernate?

Lazy loading in Hibernate means fetching and loading the data, only when it is needed, from a persistent storage like a database. Lazy loading improves the performance of data fetching and significantly reduces the memory footprint.

**Q:17 What is the difference between merge and update method?**

Ans: Use update() method when you are sure that session does not contain an already persistent instance with the same identifier, and merge() if you want to merge your modifications at any time without consideration of the state of the session.

**Q:18 How do you define sequence generated primary key in hibernate?**

Ans:   
Using <generator> tag.  
Example:-  
<id column="CUST\_ID" name="id" type="java.lang.Long">   
<generator class="sequence">   
<param name="table">SEQUENCE\_NAME</param>  
<generator>  
</id>

**Q:19 What are the different types of caches in Hibernate?**

Ans: Hibernate uses two different type of caches for objects: first-level cache and second-level cache. First level of cache is associated with Session object, while second-level of cache is associated with the SessionFactory object. By default, Hibernate uses first-level of cache on a per-transaction basis. Hibernate mainly use this cache to reduce the number of SQL queries it needs to generate within a given transaction.

**Q:20 What do you mean by Named – SQL query?**

Ans: Named SQL queries are defined in the mapping xml document and called wherever required.  
Example:  
<sql-query name = "empdetails">  
<return alias="emp" class="com.test.Employee"/>  
SELECT emp.EMP\_ID AS {emp.empid},  
emp.EMP\_ADDRESS AS {emp.address},  
emp.EMP\_NAME AS {emp.name}   
FROM Employee EMP WHERE emp.NAME LIKE :name  
</sql-query>  
Invoke Named Query :  
List people = session.getNamedQuery("empdetails")  
.setString("Deepak", name)  
.setMaxResults(50)  
.list();

Q : What is Hibernate proxy?

A proxy is returned when actually a load() method is called on a session.   
Hibernate will initially return CGLIB proxies which implement the named interface.   
The actual persistent object will be loaded when a method of the proxy is invoked.

http://www.dineshonjava.com/p/proxy-objects-and-eager-and-lazy-fetch.html#.VRHu7vmUdKA

Another good one ::::( <http://stackoverflow.com/questions/25340606/what-does-the-hibernate-proxy-object-contain>)

* Hibernate uses a proxy object to implement lazy loading. When we request to load the Object from the database, and the fetched Object has a reference to another concrete object, Hibernate returns a proxy instead of the concrete associated object.
* Hibernate creates a proxy object using bytecode instrumentation (provided by javassist). Hibernate creates a subclass of our entity class at runtime using the code generation library and replaces the actual object with the newly created proxy.

#### Explain the types of Hibernate instance states?

* Transient  
  In this state, an instance is not associated with any persistence context
* Persistent  
  In this state, an instance is associated with a persistence context
* Detached  
  This is a state for an instance which was previously associated with a persistence context an has been currently closed dissociated

#### Explain the difference between transient (i.e. newly instantiated) and detached objects in hibernate.

* Transient objects do not have association with the databases and session objects. They are simple objects and not persisted to the database. Once the last reference is lost, that means the object itself is lost. The commits and rollbacks will have no effects on these objects. They can become into persistent objects through the save method calls of Session object.
* The detached object have corresponding entries in the database. These are persistent and not connected to the Session object. These objects have the synchronized data with the database when the session was closed.

#### What is the advantage of Hibernate over jdbc?

* Developer has to write code in JDBC to map an object model's data representation to a relational data model and its corresponding database schema.   
  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.
* 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.
* With JDBC, caching needs to be manually maintained.   
  Hibernate cache is set to application work space. Relational tuples are moved to this cache as a result of query. It improves performance during multiple writes for the same data.
* 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.

### 44. Why are callback interfaces useful in Hibernate?

Callback interfaces are useful as it allows the application to receive important notification about the objects that are in execution phase. It includes the objects that are loaded, saved or deleted from the Hibernate. Callback interfaces are very important to implement the general functionality like audit records, etc. It sends a notification when any object even occurs. It allows the programmer to get the error information or exception handling can be done in a better way to notify the user on run time in case of any problem in the programming code.

### What is version checking in Hibernate ?

version checking used in hibernate when more then one thread trying to access same data.   
For example :   
User A edit the row of the TABLE for update ( In the User Interface changing data - This is user thinking time)   
and in the same time User B edit the same record for update and click the update.   
Then User A click the Update and update done. Chnage made by user B is gone.   
  
In hibernate you can perevent slate object updatation using version checking.   
  
Check the version of the row when you are upding the row.   
Get the version of the row when you are fetching the row of the TABLE for update.   
On the time of updation just fetch the version number and match with your version number ( on the time of fetching).   
  
This way you can prevent slate object updatation.

<http://www.techfaq360.com/hibernate_interview_questions.jsp?qid=341>

### How to Execute Stored procedure in Hibernate ?

Option 1:   
Connection con = null;   
  
  
try {   
con = session.connection();   
  
CallableStatement st = con   
.prepareCall("{call your\_sp(?,?)}");   
st.registerOutParameter(2, Types.INTEGER);   
st.setString(1, "some\_Seq");   
  
st.executeUpdate();   
  
Option 2:   
<sql-query name="selectAllEmployees\_SP" callable="true">   
<return alias="emp" class="employee">   
<return-property name="empid" column="EMP\_ID"/>   
  
<return-property name="name" column="EMP\_NAME"/>   
<return-property name="address" column="EMP\_ADDRESS"/>   
{ ? = call selectAllEmployees() }   
</return>   
</sql-query>   
  
code :   
  
SQLQuery sq = (SQLQuery) session.getNamedQuery("selectAllEmployees\_SP");   
  
List results = sq.list();

40.What is automatic dirty checking?

Automatic dirty checking is a feature that saves us the effort of explicitly asking Hibernate to update the database when we modify the state of an object inside a transaction.

42.What are Callback interfaces?

Callback interfaces allow the application to receive a notification when something interesting happens to an object—for example, when an object is loaded, saved, or deleted. Hibernate applications don't need to implement these callbacks, but they're useful for implementing certain kinds of generic functionality.