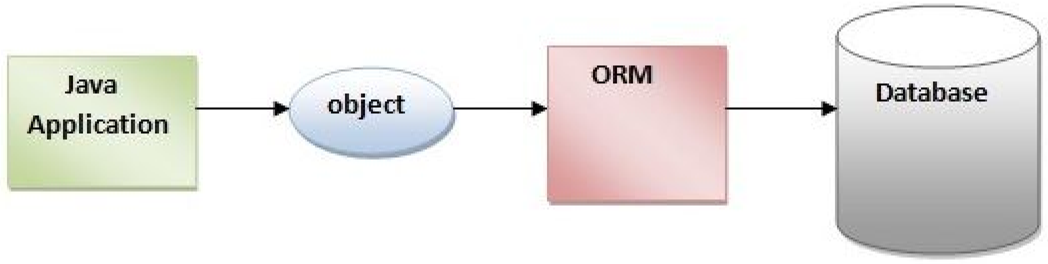
1) What is hibernate?
Hibernate is an open-source and lightweight ORM tool that is used to store, manipulate and retrieve data from the database. (ORM Object Relational Mapping)
An **ORM tool** simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



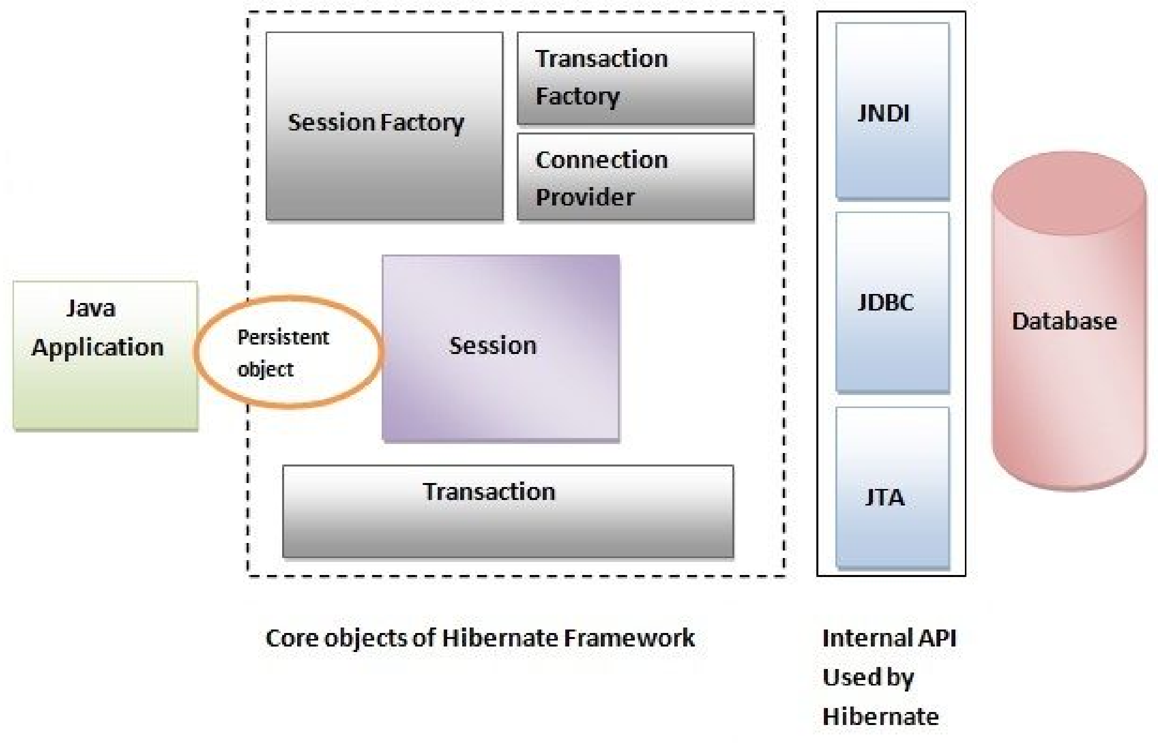
The ORM tool internally uses the JDBC API to interact with the database.

Advantages of Hibernate Framework

1) Opensource and Lightweight: Hibernate framework is opensource under the LGPL license and lightweight.
2) Fast performance: The performance of hibernate framework is fast because cache is internally used in hibernate framework.
3) Database Independent query: HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, If database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.
4) Automatic table creation: Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.
5) Simplifies complex join: To fetch data form multiple tables is easy in hibernate framework.
6) Provides query statistics and database status: Hibernate supports Query cache and provide statistics about query and database status.

---

3) Explain hibernate architecture?
Hibernate architecture comprises of many interfaces such as Configuration, SessionFactory, Session, Transaction etc.



4) What are the core interfaces of Hibernate?
The core interfaces of Hibernate framework are:

- Configuration
- SessionFactory
- Session
- Query
- Criteria
- Transaction

5) What is SessionFactory?

SessionFactory, as the name suggest, is a factory to hibernate Session objects. SessionFactory is often built during start-up and used by application code to get session object. It acts as a single data store and it's also thread-safe so that multiple threads can use the same SessionFactory. Usually, a Java JEE application has just one SessionFactory, and individual threads, which are servicing client’s request obtain hibernate Session instances from this factory, that’s why any implementation of SessionFactory interface must be thread-safe. Also, the internal state of SessionFactory, which contains all metadata about Object/Relational mapping is Immutable and can not be changed once created.

6) Is SessionFactory a thread-safe object?
Yes, SessionFactory is a thread-safe object, many threads cannot access it simultaneously.

7) What is Session?
It maintains a connection between hibernate application and database.
It provides methods to store, update, delete or fetch data from the database such as persist(), update(), delete(), load(), get() etc.
It is a factory of Query, Criteria and Transaction i.e. it provides factory methods to return these instances.

Session maintain a connection with the database and are **not thread-safe**, it means you can not share Hibernate Session between multiple threads.

8) Is Session a thread-safe object?
No, Session is not a thread-safe object, many threads can access it simultaneously. In other words, you can share it between threads.

9) What is the difference between update and merge method?

The differences between update() and merge() methods are given below.

No.	update() method	merge() method
1)	Update means to edit something.	Merge means to combine something.
2)	update() should be used if session doesn't contain an already persistent state with same id. It means update should be used inside the session only. After closing the session it will throw error.	merge() should be used if you don't know the state of the session, means you want to make modification at any time.

Let's try to understand the difference by the example given below:

SessionFactory factory = cfg.buildSessionFactory();

Session session1 = factory.openSession();

```
Employee e1 = (Employee) session1.get(Employee.class, Integer.valueOf(101));//passing id of employee

session1.close();

e1.setSalary(70000);


Session session2 = factory.openSession();

Employee e2 = (Employee) session1.get(Employee.class, Integer.valueOf(101));//passing same id

Transaction tx=session2.beginTransaction();

session2.merge(e1);

tx.commit();

session2.close();
```

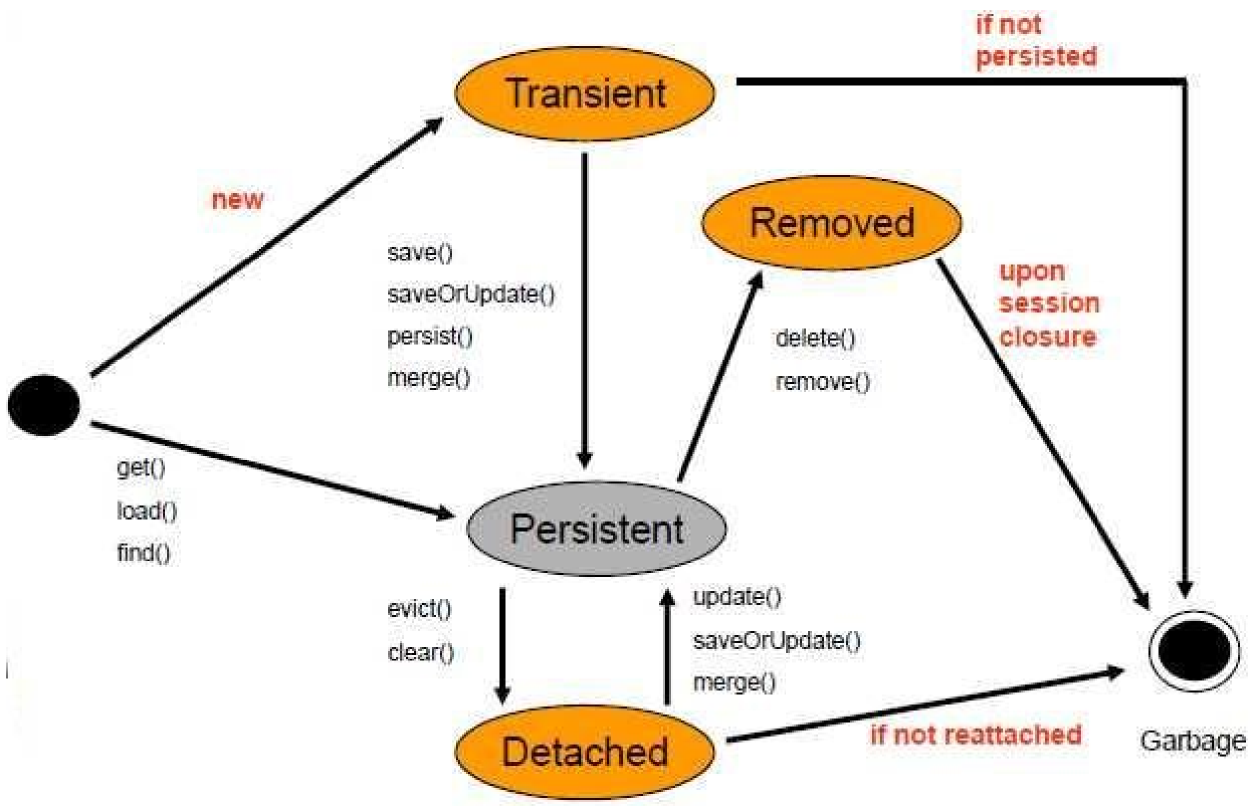
After closing session1, e1 is in detached state. It will not be in session1 cache. So if you call update() method, it will throw an error.
Then, we opened another session and loaded the same Employee instance. If we call merge in session2, changes of e1 will be merged in e2 and then merged object will be saved in database.

---

10) What are the states of object in hibernate?

There are 3 states of object (instance) in hibernate.

1. **Transient**: The object is in transient state if it is just created but has no primary key (identifier) and not associated with session.
2. **Persistent**: The object is in persistent state if session is open, and you just saved the instance in the database or retrieved the instance from the database.
3. **Detached**: The object is in detached state if session is closed. After detached state, object comes to persistent state if you call merge() or update() method.



---

13) What are the inheritance mapping strategies?

There are 3 ways of inheritance mapping in hibernate.

1. Table per hierarchy
2. Table per concrete class
3. Table per subclass

14) How to make a immutable class in hibernate?
If you mark a class as mutable="false", class will be treated as an immutable class. By default, it is mutable="true".

---

15) What is automatic dirty checking in hibernate?
The automatic dirty checking feature of hibernate, calls update statement automatically on the objects that are modified in a transaction.
Let's understand it by the example given below:

```
...

SessionFactory factory = cfg.buildSessionFactory();

Session session1 = factory.openSession();
```

```
Transaction tx=session1.beginTransaction();

Employee e1 = (Employee) session1.get(Employee.class, Integer.valueOf(101));

e1.setSalary(70000);  //state is modified

tx.commit();

session1.close();
```

Here, after getting employee instance e1 and we are changing the state of e1.
After changing the state, we are committing the transaction. In such case, state will be updated automatically, please mind that we have not called session.update() or similar method explicitly. This is known as dirty checking in hibernate. It is done with the help of persistent context where Hibernate keeps one more copy of state and compare if state is modified.

---

16) How many types of association mapping are possible in hibernate?

There can be 4 types of association mapping in hibernate.
Every example is in `<class name = "Employee" table = "EMPLOYEE">` `</class>`

1. Many to One ; using <many-to-one> tag

```
    <many-to-one name = "address" column = "address"
  class="Address" not-null="true"/>
```

2. One to one; by using <many to one > tag with unique property

```
<many-to-one name = "address" column = "address" unique="true"
      class="Address" not-null="true"/>
```

3. one to many; Using set

```
<class name = "Employee" table = "EMPLOYEE">
    <set name = "certificates" cascade="all">
      <key column = "employee_id"/>
      <one-to-many class="Certificate"/>
    </set>

</class>
```

4. Many to Many; using set

```
<set name = "certificates" cascade="save-update" table="EMP_CERT">
      <key column = "employee_id"/>
      <many-to-many column = "certificate_id" class="Certificate"/>
    </set>
```

17) Is it possible to perform collection mapping with One-to-One and Many-to-One?
No, collection mapping can only be performed with One-to-Many and Many-to-Many

18) What is lazy loading in hibernate?
An object that doesn't contain all of the data you need but knows how to get it.

Lazy loading is a design pattern commonly used in computer programming to defer initialization of an object until the point at which it is needed. It can contribute to efficiency in the program's operation if properly and appropriately used

Lazy fetching decides whether to load child objects while loading the Parent Object. You need to do this setting respective hibernate mapping file of the parent class. Lazy = true (means not to load child) By default the lazy loading of the child objects is true.
This make sure that the child objects are not loaded unless they are explicitly invoked in the application by calling getChild() method on parent.In this case hibernate issues a fresh database call to load the child when getChild() is actually called on the Parent object.
But in some cases you do need to load the child objects when parent is loaded. Just make the lazy=false and hibernate will load the child when parent is loaded from the database.
Example :
If you have a database table Employee mapped to Employee object and contains set of Addresses objects.
Parent Class : Employee class, Child class : Address Class

```
public class Employee {
 private Set address = new HashSet(); // contains set of child Address objects
 public Set getAddress () { return address; }
 public void setAddresss(Set address) { this. address = address; }
}
```

In the Employee.hbm.xml file

```
<set name="address" inverse="true" cascade="delete" lazy="false"> <key column="a_id" />
<one-to-many class="beans Address"/> </set>
```

In the above configuration. If lazy="false" : - when you load the Employee object that time child object Address is also loaded and set to setAddresss() method. If you call employee.getAdress() then loaded data returns.No fresh database call.
If lazy="true" :- This the default configuration. If you don't mention then hibernate consider lazy=true. when you load the Employee object that time child object Adress is not loaded. You need extra call to data base to get address objects. If you call employee.getAdress() then that time database query fires and return results. Fresh database call

Lazy loading in hibernate improves the performance. It loads the child objects on demand.Since Hibernate 3, lazy loading is enabled by default, you don't need to do lazy="true". It means not to load the child objects when parent is loaded.

---

19) What is HQL (Hibernate Query Language)?
Hibernate Query Language is known as an object oriented query language. It is like structured query language (SQL).
The main advantage of HQL over SQL is:

1. You don't need to learn SQL
2. Database independent
3. Simple to write query

---

20) What is the difference between sorted and ordered collection in hibernate?

A sorted collection is sorted in memory by using Java Comparator while an ordered collection uses database's order by clause for ordering. For large data set it's better to use ordered collection to avoid any OutOfMemoryError in Java, by trying to sort them in memory.

---

21) Can we make a Hibernate Entity Class final?

Yes, you can make a Hibernate Entity class final, but that's not a good practice. Since Hibernate uses a proxy pattern for performance improvement in the case of the lazy association, by making an entity final, Hibernate will no longer be able to use a proxy, because Java doesn't allow extension of the final class, thus limiting your performance improvement options. Though, you can avoid this penalty if your persistent class is an implementation of an interface, which declares all public methods defined in the Entity class.

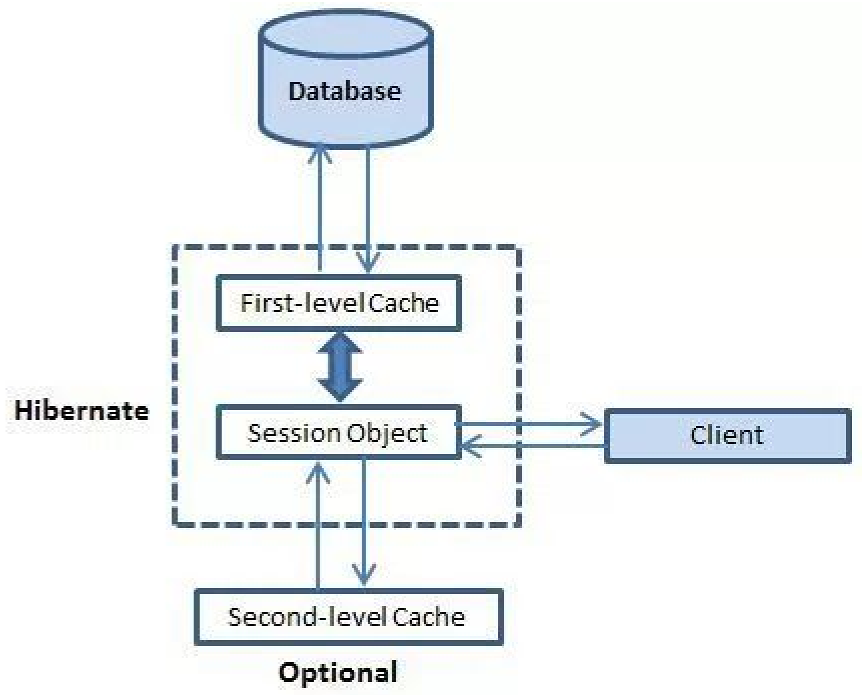
---

22) What is First level cache in Hibernate?

The first-level cache is the Session cache and is a mandatory cache through which all requests must pass. The Session object keeps an object under its own power before committing it to the database.If you issue multiple updates to an object, Hibernate tries to delay doing the update as long as possible to reduce the number of update SQL statements issued. If you close the session, all the objects being cached are lost and either persisted or updated in the database.

---

23) What is second level Cache and what are it's types?



Hibernate second level cache uses *a common cache for all the session object of a session factory*. It is useful if you have multiple session objects from a session factory. **SessionFactory** holds the second level cache data. It is global for all the session objects and not enabled by default.
Different vendors have provided the implementation of Second Level Cache.

1. EH Cache
2. OS Cache
3. Swarm Cache
4. JBoss Cache

Each implementation provides different cache usage functionality. There are four ways to use second level cache.

1. **read-only:** caching will work for read only operation.
2. **nonstrict-read-write:** caching will work for read and write but one at a time.
3. **read-write:** caching will work for read and write, can be used simultaneously.
4. **transactional:** caching will work for transaction.

The cache-usage property can be applied to class or collection level in hbm.xml file. The example to define cache usage is given below:

```
<hibernate-mapping>
        <class name="Employee" table="EmployeeTable">
        <cache usage="read-only" />
        <id name="id">
                <generator class="native"></generator>
```

```
            </id>
            </class>
    </hibernate-mapping>
```

We need to make changes in Hibernate Configuration file also

```
<!-- Enable the second-level cache -->
<property name="cache.use_second_level_cache">true</property>
<property name="cache.provider_class">org.hibernate.cache.EhCacheProvider</property>
```

---

24) What is Query Cache?

Query Cache is used to cache the results of a query. When the query cache is turned on, the results of the query are stored against the combination query and parameters. Every time the query is fired the cache manager checks for the combination of parameters and query. If the results are found in the cache, they are returned, otherwise a database transaction is initiated. As you can see, it is not a good idea to cache a query if it has a number of parameters, because then a single parameter can take a number of values. For each of these combinations the results are stored in the memory. This can lead to extensive memory usage.

To turn it on we make following changes in Hibernate Configuration files
<prop key="hibernate.cache.use_query_cache">true</prop>

and

Query query = session.createQuery("FROM EMPLOYEE");
query.**setCacheable**(true);
List users = query.list();

25) What is the difference between get and load method ?

Main **difference between get() vs load method** is that get() involves database hit if object doesn't exists in Session Cache and returns a fully initialized object which may involve several database call while load method can return proxy in place and only initialize the object or hit the database if any method other than getId() is called on persistent or entity object. This lazy initialization can save couple of database round-trip which result in better performance.

get()	**load()**
Returns **null** if object is not found.	Throws **ObjectNotFoundException** if object is not found.
get() method always **hit the database**.	load() method **doesn't hit** the database.

It returns real object **not proxy**.	It returns **proxy object.**
It should be used if **you are not sure** about the existence of instance.	It should be used if **you are sure** that instance exists.

25) What is the difference between save, persist and saveOrUpdate methods in Hibernate?

1. Save() method stores an object into the database. It will Persist the given transient instance, first assigning a generated identifier. It returns the id of the entity created. Used for first time insert.
2. SaveOrUpdate() calls either save() or update() on the basis of identifier exists or not. e.g if identifier does not exist, save() will be called or else update() will be called.
3. Persist does not return anything after saving; it's return type is void.

26) What are different type of cascading?

Assume you have a stock class which may have many stockDailRecords in a relation; so it is one to many mapping which is achieved with the help of set collection.

1. **Save-Update**

The cascade="save-update" is declared in 'stockDailyRecords' to enable the save-update cascade effect.

```
<!-- Stock.hbm.xml -->
<set name="stockDailyRecords" cascade="save-update" table="stock_daily_record"...>
    <key>
        <column name="STOCK_ID" not-null="true" />
    </key>
    <one-to-many class="com.mkyong.common.StockDailyRecord" />
</set>
```
I.e saving stock will automatically save related stockDailyRecords

2. Delete

The cascade="delete" is declared in 'stockDailyRecords' to enable the delete cascade effect. When you delete the 'Stock', all its reference 'stockDailyRecords' will be deleted automatically.

```
<!-- Stock.hbm.xml -->
<set name="stockDailyRecords" cascade="delete" table="stock_daily_record" ...>
    <key>
        <column name="STOCK_ID" not-null="true" />
    </key>
    <one-to-many class="com.mkyong.common.StockDailyRecord" />
</set>
```
I.e deleting stock will automatically delete related stockDailyRecords

3. Delete Orphan
The cascade=”delete-orphan” is declared in ‘stockDailyRecords’ to enable the delete orphan cascade effect. When you save or update the Stock, it will remove those ‘stockDailyRecords’ which already mark as removed.

```
<!-- Stock.hbm.xml -->
<set name="stockDailyRecords" cascade="delete-orphan" table="stock_daily_record" >
    <key>
        <column name="STOCK_ID" not-null="true" />
    </key>
    <one-to-many class="com.mkyong.common.StockDailyRecord" />
</set>
```

Copy

```
StockDailyRecord sdr1 = (StockDailyRecord)session.get(StockDailyRecord.class,
                                          new Integer(56));
StockDailyRecord sdr2 = (StockDailyRecord)session.get(StockDailyRecord.class,
                                          new Integer(57));

Stock stock = (Stock)session.get(Stock.class, new Integer(2));
stock.getStockDailyRecords().remove(sdr1);
stock.getStockDailyRecords().remove(sdr2);

session.saveOrUpdate(stock);
```

27) What is named query?

Named queries are SQL queries which are defined in mapping document using <query>, <sql-query> tag and called using Session.getNamedQuery() method. Named query allows you to refer a particular query by the name you provided, by the way, you can define named query in hibernate either by using annotations or XML mapping file. @NameQuery is used to define single named query and @NameQueries is used to define multiple named query in hibernate

In hbm.xml file

```
<hibernate-mapping>

        <query name="HQL_GET_ALL_EMPLOYEE">from Employee</query>
        <sql-query name="SQL_GET_ALL_EMPLOYEE">
                <![CDATA[select emp_id, emp_name, emp_salary from Employee]]>
        </sql-query>

</hibernate-mapping>
```

***CDATA : **CDATA** is defined as blocks of text that are not parsed by the parser, but are otherwise recognized as markup.A try to parse CDATA might result in parsing error and stop the program execution.*

```
In main class file
//HQL Named Query Example
        Query query = session.getNamedQuery("HQL_GET_ALL_EMPLOYEE");
        List<Employee> empList = query.list();
        for (Employee emp : empList) {
                System.out.println("List of Employees::" + emp.getId() + ","
                                + emp.getAddress().getCity());
        }
```

28) Give one simple example of Hibernate.