1. what is diff between jdbc and hibernate. and why we are used hibernate.  
-------------------------------------------------------------------------------  
JDBC  
--------  
-> it is database dependent.  
-> jdbc does not send the data in the form of object. it will allow to send the data in form of text or value.  
-> jdbc is an api provided by third party vendor. its throw sql exception i.e checked exception so we need to write lot of try catch block.its a boiler plate code.  
-> jdbc does not support cache mechanism so performance of application is low.  
-> jdbc code is very tightly coupled with the application.  
-> jdbc does not support relationships.

Hibernate  
--------------  
-> hibernate is a database independent so we can run any platform.  
-> but in real-time we need to transfer the data in the form of objs. in hibernate we can send the data in the form of object.  
-> hibernate is an ORM framework. its throw uncheked exception so we no need to write try catch an throws exception.hibernate builts in transaction managemnet  
system.it removes the usage of try catch blocks.  
-> hibernate support caching mechanism .it improves the performance of the application for efficient data retrieval.  
-> hibernate support jpa annotaions. so it removes lot of boiler plate code that comes jdbc api the code looks like more readable and clears.  
-> its support inheritance or association relationship.

2. what is ORM (object relational mapping).  
------------------------------------------------  
-> Hibernate is the most-popular persistence framework and ORM tool for Java Applications.  
-> ORM (Object/Relational Mapping) is a methodology where objects in Java Applications are persisted transparently in the relational database tables.  
-> ORM stands for Object Relational mapping. It is programming paradigm which is used to persist java objects to database tables.

3. what is session.  
-----------------------------  
-> session is an interface present in "org.hibernate.package".  
-> it is used to perform db operation. session provide methods to perform create, read, update, and delete operations for a persistent obj.  
-> we can execute hql quries, sql native quires and create criteria using session obj.

4. what is Session Factory.  
-----------------------------------  
-> session factory is an interface present in "org.hibernate package".  
-> session factory is a state which contain mapping file & configuration file information and create the connection to perform the db operation.  
-> Session Factory is a heavy weight object and it should be created one per database. SessionFactory object is shared by multiple sessions.  
-> session factory is a heavy weight obj that has to be created only once per application.  
-> when u have multiple db in your application you should create multiple session factory objects. eg if u are using two db called mysql & oracle in your  
hibernate application then u need to build 2 session factory objects.  
sessionfactory factory = newconfiguration().buildsessionfactory();.

5. what is diff between session and session factory.  
---------------------------------------------------------  
-> session  
--------------  
-> session is not a threadsafe.  
-> it is mutable.  
-> it is a lightweight.  
-> performance wise it is high compare to session factory.

sessionfactory  
--------------------------  
-> sessionfactory is threadsafe so, many thread can access it concurrently or simulteniously.  
-> it is immutable and it will be created as singleton while the server initialize itself.  
-> it is heavy weight becoz it maintain datasource, mapping file, and configuration file information.  
-> performance wise it is low compare to session.

why should we need to make sessionfactory obj of hibernate as singleton.  
----------------------------------------------------------------------------------  
-> in hibernate a sessionfactory obj is the only one heavy obj becoz it stores configuration data and all the mapping data of a project.  
-> if our project is a desktop application, then there is no need to make sessionfactory obj as a singleton, becoz only one user can access the  
application at a time.  
-> if a proj is a distributed application, it means it is a web application, remoting application, then at a time multiple clients can send a request  
to the server application. if multiple sessionfactory objets are creted then the burden on the server will be increased. so we should make session  
factory obj as a singletone.  
-> by default sessionfactory obj of hibernate is not a singleton, it is the programmer responsibility to make it as a singleton.

public class HibernateUtil{  
private static SessionFactory factory;  
private HibernateUtil()  
{}  
public static synchronized SessionFactory getSessionFactory{  
if(factory==null)  
{  
factory=new configuration().configure("hibernate.cfg.xml").buildSessionFactory();  
}  
return factory;  
}  
}

6. what is dialect in hibernate. and how it works.  
------------------------------------------------------------  
-> dialect is a simple java class present in org.hibernate.dialect package.  
-> every configuration file contains dialect which is very important while performing the db operation.  
-> dialect class is used to convert hql queries into db specific queries.  
-> if u want to shift from one db to another db just change the dialect class name and connection details in hibernate cfg file.

Lazy loading & Eager loading.  
---------------------------------------  
-> it is also an important feature of hibernate which improves performance of the application by reducing network round trips between java application and database .  
-> in lazy loading hibernate doesnt load an object from database immediately. an object will be loaded on demand basis.  
-> if an object is immediately loaded then that obj is called as eager/early loading.  
-> hibernate lazy loading is true by default.  
-> since hibernate 3 lazy loading is by deafult enabled so that child objs are not loaded when parent is loaded.  
Fetch=FetchType.LAZY.  
-> in lazy loading, hibernate creates a proxy object returns proxy obj to the application.  
-> when application is accessing the proxy object, then internally the data is selected from the database.  
-> with lazy loading feature, no of trips with the database are going to be reduced and performance will be improved.  
Lazy loading means when you load parent obj, child objects won’t get loaded until requested.

what is Caching in hibernate.  
---------------------------------------  
-> it is the most important features of hibernate.  
-> caching is a mechanism to store the object which are loaded from the databse.  
-> Caching is facility provided by ORM frameworks which help users to get fast running web application.  
-> Hibernate also provide this caching functionality, in two layers.

the main advantage of caching is  
--------------------------------------  
-> it is reduce/save the number of round trips between java application and the database and it improves the performance of an application.  
-> when an application wants an object from database, then hibernate looks for that object at level 1 cache, if not found then it looks for that obj at level2 cache. even though if it is not found, then only it goes for database.  
-> hibernate maintains cache at two levels hence hibernate gives better performance.  
-> cache is a temperory(buffer) memory that is allocated to RAM.  
-> in hibernate cache is managed by two objects.  
1.> session object  
2.> session factory object.  
-> a cache managed by session obj is called as level 1 cache or local cache.  
-> a cache managed by session factory obj is level 2 cache.

7. what is first level cache .  
------------------------------------------  
-> first level cache is automatically created by hibernate when a session is opened and it will be automatically closed when a session will be closed.  
-> by default, for each hibernate application the first level cache is bydefault enabled and as a programmer we are not able to enable or disable first level cache.  
-> first level cache is associated with "session" object and other session object in application can not see it.  
-> first level of cache has a session scope.  
-> first level cache session is not sharable with other session, so it is called local cache.  
-> as a programmer we don't need to add any special tags in .hbm file or in .cfg file.  
-> first level cache is not in the hands of a programmer. it is in hands of hibernate only.  
-> each session will have its own cache. a session cannot read an obj from another session cache.

8. what is second level cache.  
---------------------------------------  
-> second level cache is introduced in hibernate 3.0 and it is associated with session factory object.  
-> and its scope is application level. its called global cache.  
-> second level cache is created in session factory scope and it is sharable with other session, so it is called global cache.  
-> when we are loading an object from the db, then hibernate will search for the obj in the first level cache, if exist then it is loaded from first level cache.  
-> if not exist, then hibernate will search for second level cache if exist then the object will be copied to first level cache .  
-> there are multiple second level cache s/w are available but widely used second level cache is ehcache(EasyHibernateCache), os cache, jboss cache.  
-> second level cache is not by default enabled, we need to enabled by adding following two properties hibernate cfg.xml file.  
-> add provider class in configuration file.  
<property name:"hibernate.cache.providers\_class>  
org.hibernate.cache.EHcacheProvider  
</property>  
-> add cache tag in mapping of pojo class.  
<cache usage="read-only"> or read-write value.  
-> create a xml file called ehcache.xml and place in classpath.  
ehcache.xml it contains.  
1.> behaviour of cache  
2.> time setting.  
3.> update setting.  
4.> life time of pojo class  
1>. cache:use\_second\_level\_cache.  
2>. cache:provider\_class.  
-> in order to do setting for a second level cache, we need to create ehcache.xml along with hibernate jars, we need to add a third party jar ehcache-1.2.3 jar.  
-> hibernate has provided, provider classes to get the cache from third party cache providers.  
1> org.hibernate.cache.ehcache.providers.  
2> org.hibernate.cache.oscache.providers.  
-> while performing second level cache we should also choose caching strtegy they are.  
1.> read only.  
2.> read/write.  
3.> non strict read/write.  
-> we should know different strategies for caching an object.  
-> 1.> Read Only: This caching strategy should be used for persistent objects that will always read but never updated. It’s good for reading and  
caching application configuration and other static data that are never updated. This is the simplest strategy with best performance because  
there is no overload to check if the object is updated in database or not.  
-> 2.> Read Write: It’s good for persistent objects that can be updated by the hibernate application. However if the data is updated either through  
backend or other applications, then there is no way hibernate will know about it and data might be stale. So while using this strategy,  
make sure you are using Hibernate API for updating the data.

configuring of ehcache  
--------------------------  
-> in hibernate cfg.xml, we need to configure following properties.

<property name= "cache.provider\_class"> org.hibernate.cache.ehcacheprovider </property>  
<property name= "cache.use\_secondlevel\_cache"> true </propery>

ehcache.xml  
-----------------  
<ehcache>  
<default cache maxElementMemory="100" eternal="false" timeToIdleSeconds="120"timeToLiveseconds="200"/>  
<cache name="Employee" maxElementInMemory="10" eternal="false"timeToIdleSeconds="8" TimeToLiveSeconds="300"/>  
<ehcache>  
-> in the above xml IdleSeconds indicates waiting time before an object is going to be removed from cache.  
-> LiveSeconds indicates a complete lifetime of an object, before it is going to be deleted from cache.  
-> to make IdleSeconds and LiveSeconds as working, we need to disable eternal by using eternal="false".  
-> in ehcache.xml,if eternal="true" then we should not write timeToIdleSeconds,TimeToLiveSeconds, hibernate will take care by about those values.  
-> so,if u want to give values manually better use eternal="false" always, so that we can assign values into timeToIdleSeconds,TimeToLiveSeconds manually.

9. what is diff between first level cache and second level cache.  
----------------------------------------------------------------------------  
-> first level cache  
-----------------------------  
-> first level cache is associated with session obj.  
-> first level cache is by default enabled.  
-> first level cache scope is session level.  
-> first level cache is not sharable.  
-> first level cache is avilable only untill the session is opened, once the session is closed. the first level cache is destroyed.

second level cache  
--------------------------------  
-> second level cache is associated with session factory obj.  
-> it is optional. if u want then enabled hibernate.cfg.xml.  
-> second level cache scope is application level.  
-> second level cache is sharable.  
-> second level cache is avilable through the application life cycle. it is only destroy when your application is restart.

what is mapping file how to write this.  
------------------------------------------------  
-> mapping file is a mechanism of placing an object properties into column of a table.  
-> a java application can have multiple classes and a database can have multiple tables.  
-> hibernate knows how to store an object and how to read it, but it doesn't know which java class object need to be persisted in which table of database.  
-> it is the programmer responsbility to tell hibernate that which java class obj need to be stored in which table. to pass this information to hibernate ,  
as a programmer we need to construct the mapping files. we can map multiple classes in a single hbm file.  
-> mapping can be done by using annotation also. if we use annotations for mapping then we no need to write mapping file. from hibernate 3.0 version its support.  
-> each hibernate mapping file must contain one <id> tag property.  
-> in mapping file class names & property name are case sensetive.  
-> but table name and column name are not case sensative.  
-> we can map multiple classes in a single hbm file.  
<hibernate-mapping>  
<class name="Employee" table="emp">  
<id name="emp\_no" column="empno"/>  
<property name="emp-name" column="ename"/>  
<property name="emp-sal" column="esal"/>  
<property name="dept-number" column="deptno"/>  
</class>  
<hibernate-mapping>

what is configuration file and how to write this.  
------------------------------------------------------  
-> It is used to bootstrap hibernate and it is used to locate to hibernate mapping file.  
-> configuration file is an xml file, this configuration file contains 3 types of information.  
1> connection properties.  
2> hibernate properties.  
3> mapping resource (file name).  
-> we can create one configuration file for each database. it means the number of configuration files in a project depends on number of databases.  
-> suppose if we want to connect with 2 database like oracle, mysql then we must create 2 configuration file.  
-> in hibernate, it internally opens a connection with the database and also it closes automatically.  
-> as a programmer we need to provide connection properties to hibernate through configuration file,  
<hibernate-configuration>  
<session-factory>  
<property name="connection.driver\_class">driver class name </property>  
<property name="connection.url"> url</property>  
<property name="connection.username">username</property>  
<property name="connection.password>password</property>  
<propert name="show-sql">true</property>  
<mapping resource="Employee.hbm.xml/>  
</session-factory>  
</hibernate-configuration>

10. what is cascade in hibernate.  
-----------------------------------------  
-> cascade is an attribute. it is a mandatory when we apply parent child relationship between the objects.  
-> when we have relationship between entities, then we need to define how the diff operations will affect the other entity.  
-> this is done by cascading and there are diff types of it.  
-> cascade attributes takes multiple values none, all, save-update, all-delete-orphan.  
-> whenever we perform any operation on parent table it perform the operation on child table also.  
-> by default values of cascade="none" means no operation will tranfer to the child class.  
-> eg if we apply insert(update or delete) operation on parent class obj, then child class obj will also be stored into the database.  
-> eg if we apply insert(update or delete) on parent class object will not be effected,if cascade="none".  
import.hibernate.annotations.cascade;  
@Entity  
@Table(name="EMPLOYEE")  
public class Employee {  
@OneToOne(mappedBy="employee")  
@Cascade(value=org.hibernate.annotations.CascadeType.ALL)  
private Address address;  
}

what is orphan record  
--------------------------------  
-> it is a record in child table but it doesnt have association with its parent in the application.  
-> in an application, if a child record is removed from the collection and if we want to remove that child record immediately from the database, then we need  
to set the cascade="all-delete-orphan".

11. what is inverse in hibernate.  
-----------------------------------------  
-> default value of inverse="false".  
-> if inverse="false" hibernate will not check the bi-directional relationships between the tables. in this scenario if we try to send multiple insert queries, and update queries.  
-> if inverse="false", parent class is responsible for saving/updating the child and its relationships.  
-> if the inverse is="true" and associated subclass is responsible for saving/updating itself.

NOTE  
---------  
-> An inverse keyword is always used with the one-to-many and many-to-many relationships.  
-> it doesnt work with many to one relationships.

12. what is lifecycle of hibernate or state of an object.  
------------------------------------------------------------------------

1. transient state.  
-----------------------------  
-> in programming world every object has life cycle after birth and before death.  
-> there are three states in hibernate obj will perform the persistency.  
-> when obj has been created then that state is called as transient state.  
-> if we perform any operation also it will not effect into the database or session cache.  
-> If object is in transient state, it means it was never associated with session and just created.

2. persistent state.  
-----------------------  
-> an object is said to be a persistent state, when it is associated with session as well as obj present in database to perform operation.  
-> When an object is saved to database using hibernate session via Session.save() or Session.persist() method, then this object is called to be in persistent state.  
-> there are several method provided by hibernate to get obj into persistency state eg save, persist, saveorupdate.

3. Detached state.  
-------------------------  
-> an obj is said to be detached state, when the obj is not associated with session but present in the database.  
-> there are several method which can easily detached the obj from persistency state to detached state. eg-close, evict, clear.  
-> If object is in detached state, it means session is closed and object is no more part of session.  
-> If you call merge or update, object goes back to persistent state.

13. what is HQL.(hibernate Query language).  
-------------------------------------------------  
-> by calling save(), update(), delete() we can perform bulk operation.  
-> we can perform CURD operation on a single obj at a time. if we want to perform CURD operations on multiple objects at a time, we use bulk operation  
technique of hibernate at a time. ex- HQL, criteria, Native Sql.  
-> hibernate has introduce its own query language with the name of HQL.  
-> Hql is a database independent query language. it means we no need of changing a query while connecting with another database.  
-> HQL is an Object-Oriented Query language, simillar to database SQL. But instead of working with tables and columns, HQL uses Java Objects to manipulate queries.  
-> hql doesn't provide any burden on the developer, becoz its looks like simillar to sql only. so it is easy to learn.  
-> a diff between sql and hql is, sql commands are database dependent but hql commands are database independent.  
-> to construct Hql queries, we use variable names in place of column names and class names in places of table names.  
-> we call hql as object oriented form of sql.  
-> HQL is an Object-Oriented Query language, simillar to database SQL. But instead of working with tables and columns, HQL uses Java Objects to manipulate queries.  
-> HQL supports:  
– FROM Clause.  
– AS Clause.  
– SELECT Clause.  
– WHERE Clause.  
– ORDER BY Clause.  
– GROUP BY Clause.  
– Using Named Paramters.  
– UPDATE Clause.  
– DELETE Clause.  
– INSERT Clause.  
– Aggregate Methods.  
– Pagination using Query.  
-> using hql, we can perform both select and non-select operations on database.  
eg. sql > select \*from emp  
hql > from Employee e  
-> in hibernate, reading a complete row is called reading a complete entity.  
-> to load a complete entity, we need to begin hql command with from keyword.  
ex. sql > select empno, sal from emp  
hql > select e.employeeNumber,  
e.employeeSalary from Employee e  
-> reading the values of a specific columns is called reading partial entity.  
-> in hibernate, to read a partial entity, a query begins with select keyword.  
-> if we want to execute hql command, first we need a query object.

ex 1. Query query=session.createQuery("from Employee e");  
List list=query.list();  
Iterator it=list.iterator();  
while(it.hasNext())  
{  
Employee e=(Employee)it.next();  
}  
ex 2. Query query= session.createQuery("select e.employeeNumber, e.employeeSalary from Employeen e");  
List list=query.list();  
Iterator it=list.iterator();  
while(it.hasNext())  
{  
Object[] obj=(Object[]it.next();  
sopln(obj[1]);  
}  
Hql for non-select operaions  
-------------------------------------  
-> update and delete operations of hql are simillar to sql, but insert operation is different.  
-> in hql, insert operation is used to copy the records from one table to another table.

ex. sql > update emp set sal = 9000 where deptno= 10.  
Hql > update Employee e set e.employeeSalary=9000 where e.deptNumber=10.  
ex.2  
sql> delete from emp where sal>1000  
hql> delete from employee e where e.employeeSalary >1000.

what is Criteria in hibernate.  
---------------------------------------  
-> It is an alternative to HQL, it is very useful for the search query involving the multiple conditions.  
-> we cannot use criteria to run update or delete quries or any ddl statemnts. its only used to fetch the results from the database using more obj oriented approch.  
-> we can read the same output from a database by executing the diff sql commands.  
for eg. select \*from emp;  
select empno, ename, esal from emp;  
-> the above two sql commands will return same output. but second query gives better performance.  
-> while reading the data from the database, tuned queries are important to improve the performance.  
-> as a java developer, creating tuned queries will increase the burden on the developer.  
-> to decrease the burden hibernate has provided criteria API, it internally creates tuned queries and execute them on the database. so a developer is no  
need to prepare tuned quries explicitly.

Diff between getOpen session and getCurrent session.  
-------------------------------------------------------------  
-> getOpen session  
-----------------------------  
-> It always create new Session object.  
-> You need to explicitly flush and close session objects.  
-> In single threaded environment, It is slower than getCurrentSession.

getCurrent session.  
---------------------------  
-> It creates a new Session if not exists , else use same session which is in current hibernate context.  
-> You do not need to flush and close session objects, it will be automatically taken care by Hibernate internally.  
-> In single threaded environment , It is faster than getOpenSession.

15. what is diff between session.Get() method and session.Load() method.  
--------------------------------------------------------------------------------------  
-> get() and load() both are session class method used for retrieving the data from the database.  
-> get() and load() return the data in the object format.

get()  
----------  
-> if the record is not present in the database then get() always returns null.  
-> get() not create proxy obj its always return actual object.  
-> get() always hits the database.  
-> performance wise get() is not good.  
-> get() is eager initializer, get() reads or loads an obj early. so it is called early loading.  
-> If you are not sure if object with id exists or not, you can use get.

load()  
-------------  
-> if the record is not present in the database then it returns object not found exception.  
-> load() returns proxy objects and loads data only when it is actually required . so load() is better becoz its support lazy loading.  
-> load() doesnt hits the database every time.  
-> performance wise it is better to get().  
-> load() lazy initializer.load() reads or loads an object lazely.  
-> If you are sure about existence of object, you can use load.

16. what is session.save() and session.persist().  
-----------------------------------------------------------  
-> save() and persist() both method which we are mainly used for saves the object in the database.

save()  
-------------  
-> save() method will save an obj to the database and returns the id of the saved objects(primary key) in the form of serializable type.  
-> save() can be used inside or outside the transaction boundries. save() is not fit for long time running transaction.  
-> its take more time to execute.  
-> save() method return type is serializable.

persist()  
------------------  
-> save/persist both the methods we can use to save/store an obj to database.  
-> persist() will save an obj into database but it doesnt return id of saved obj. it doesnt return anything. its return type is void.  
-> persist() can be used only within the boundary of transaction. persist() is suitable for long time running transaction. It saves data when flush is called.  
-> persist() takes less time to execute.  
-> persist() method return type is void.

17. what is session.upadate().  
----------------------------------------  
-> update() just update the record & throws hibernate exception. if record is not avialable to update.  
-> this method doesnt return anything becoz its return type is void.  
-> this method cannot update primary key value.

18. what is session.save or saveorupadate().  
---------------------------------------------  
-> save() perform only insert but saveorupdate() update the record if record is avialable otherwise insert the record in the database table.  
-> this method doesnot return anything becoz its return type is void.  
-> this method we can update primary key column value.

save -> Save stores object in database. It generates id for the object and returns it. If object already exists in database, it will throw an error.

saveorUpdate -> SaveOrUpdate method save the object if id does not exist. If it exists , it calls update method.

Merge()  
----------------  
-> merge() update the record if record is avialable otherwise insert the record in database table. it is used to add a specified obj to the first level cache.  
-> through this method we can update primary key column value also.

Hibernate jpa annotation  
---------------------------------  
1> @Entity  
-------------------  
-> it is used to mark the class as persistent java class.

2> @Table  
-------------------  
-> it is used to provide the details of the table.

3> @Id  
-----------  
-> it is used to define the primary key.

4> @GeneratedValue  
-------------------------  
-> it is used to define the primary key generation strategy.

5> @Column  
-----------------  
-> it is used to define the properties of the column that will be mapped to the annotated field. you can define several properties like name, length, nullable etc.

6> @OneToOne  
---------------------  
-> @OneToOne annotation is used to create one to one relationship between Country and Capital entities.

7> @joinColumn  
------------------------  
-> it is used to specify a mapped column for joining an entity association.

8> @Inheritance  
------------------------  
-> For implementing inheritance in hiberante, @Inheritance annotation is used.It defines inheritance strategy to be implement for entity class hierarchy.  
-> For one table per class hierarhcy,we have used Single\_Table as inheritance strategy.This annotation is defined at root level or sub hierarchy level  
where different strategy is to be applied.

9> @DescriminatorColumn  
--------------------------  
-> This annotation is used to define discriminator column for Single\_Table and joined strategy.It is used to distinguish between different class instances.  
-> This annotation is defined at root level or sub hierarchy level where different strategy is to be applied.  
-> If @DiscriminatorColumn annotation is not specified,then hibernate will create a column named as “DType” and DiscriminatorType will be string.

10> @DescriminatorValue  
------------------------------  
-> This annotation defines value in discriminator column for that class.This can only be applied on entity concrete class.

why relationships in database.  
--------------------------------------  
-> we can store our application data in a database table to make it as a persistent data.  
-> for eg. consider we have an application which contains doctors and patients data and we can store that data in single table. then data redundancy problem arises.  
-> to reduce the data redundancy, we need to store in a two different tables.  
-> in order to get the relation between the data, we also need to take the column of one table as a column in another table.  
-> in hibernate application, if we create a single pojo class and if we set the data to that obj, then there is a chance of getting data redundancy.  
-> data redundancy means, some data in multiple objs of a pojo class can be duplicated.  
-> in order to avoid (or reduce), we divide properties of one class into(multiple) two classes and then we apply a relationship between objs of the two classes.  
-> in hibernate, we can apply four types of relationships between pojo classes.

Association relationship  
-------------------------------  
-> in hibernate to put relations between entities i.e one obj with multiple objs we use association relationship.  
-> if we want to implement association relationship there must be foriegn key relationship between child to parent.

the main advantage of association relationship is  
----------------------------------------------------------  
-> to perform operations on one object we can transfer that operation on another obj.  
-> e.g if we want to retrieve the data from more than one table complusory we put assosiation in hibernate.  
-> to create assosiation relationship with object to object complusory their a must be foreign key relationship in the database table.

there are 4 types of Association in hibernate.  
----------------------------------------------------  
one-to-many relationship  
------------------------------  
-> in this relationship one parent has multiple child object.  
-> one user has multiple task.  
-> country and state. One Country can have n number of states. and team and player one team has multiple players. customer and items.  
-> to create this kind of association relationships in hibernate we need to perform two modification.  
1.> in parent pojo class we must take child class as a collection properties. eg set properties, list properties, map properties.  
2.> in parent class mapping file we must take one coressponding collection properties to map this child properties.

many-to-one relationship  
---------------------------------  
-> multiple tasks for one user.

many-to-many relationship  
----------------------------------  
-> Lets take example of Country and Language. One Country can have n number of languages and one language can be spoken by n number of countries.

Inheritance Mapping (or) inheritance Strategies  
---------------------------------------------------------  
-> through inheritance we can save both parent object and child obj in to the db.  
-> in a hibernate application, if there are multiple POJO classes and if they have common properties, then to get reusability we apply inheritance.  
-> common properties, we separate and we create in super class and we extend that super class to multiple sub classes.

CreditCard.java  
----------------------

public class CreditCard {  
private int paymentID;  
private double amount;  
private Date paymentDate;  
private int cardNumber;  
private String cardType;

getters & setters  
}

Cheque.java  
-----------------

public class Cheque {  
private int paymentID;  
private Date paymentDate;  
private int chequeNumber;  
private String chequeType;

getters & setters  
}  
-> in the above two POJO classes, there are three common properties. so in order to get Resuability, we apply inheritance by separating common properties  
to a super class (payment) like this following.

Payment.java  
------------------  
public class Payment {  
private int paymentID;  
private double amount;  
private Date paymentDate;  
getters & setters  
}  
CreditCard.java  
------------------  
public class CreditCard extends Payment {  
private int cardNumber;  
private String cardType;

getters & setters  
}  
Cheque.java  
---------------

public class Cheque extends Payment {  
private int chequeNumber;  
private String chequeType;

getters & setters  
}

-> hibernate has provided three inheritance strategies, to map the classes of hierarchy to database tables.

1.> table per class:  
--------------------------------  
-> in this hierarchy we store both parent data and child data in a single data.  
-> we need to choose this strategy, if we want to map all the classes of hierrachy to a single table of database.  
-> through descriminator column we can identify which employee it is.  
-> descriminator column help us to is identified which child is.

payment payment (single table)  
/\  
credit cheque  
card

2.> table per concrete class  
--------------------------------------  
-> if we want to map each concrete class hierarchy to a separate table of database then we need to choose this strategy.

payment credit card (separte table) cheque (separate table)  
/\  
credit cheque  
card

3.> table per concrete class  
----------------------------------  
-> if we want to map each class of hierarchy to a separate table of database, then we need to choose this strategy.

payment payment(separate table) credit card(separate table) cheque(separate table)  
/\  
credit cheque  
card

Hql queries.  
--------------------  
1.> FROM Clause - FROM clause is used to load all objects into memory.  
Example:  
String hql = "FROM Customer";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
List<Customer> custList = query.list();  
String hql = "FROM Customer";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
List<Customer> custList = query.list();

2.> AS Clause - AS clause is an optional keyword, used to create an aliases for a classes in HQL queries.  
example:  
String hql = "FROM Customer AS C";

// Above statement is the same with statement: String hql = "FROM Customer C";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
List<Customer> custList = query.list();  
String hql = "FROM Customer AS C";  
// Above statement is the same with statement: String hql = "FROM Customer C";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
List<Customer> custList = query.list();

3.> SELECT Clause - SELECT clause is used to obtain few properties of returned objects.

String hql = "SELECT C.firstName, C.age FROM Customer C";  
Query query = session.createQuery(hql);  
List<Object[]> objectList= query.list();  
String hql = "SELECT C.firstName, C.age FROM Customer C";  
Query query = session.createQuery(hql);  
List<Object[]> objectList= query.list();

4.> WHERE Clause - WHERE clause is used to narrow selected objects which must meet some conditions.

String hql = "FROM Customer C WHERE C.age = 20";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
List<Customer> custList = query.list();  
String hql = "FROM Customer C WHERE C.age = 20";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
List<Customer> custList = query.list();

5.> ORDER BY Clause - ORDER BY clause is used to sort HQL query’s results.

String hql = "FROM Customer C WHERE C.age > 25 ORDER BY C.id DESC";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
List<Customer> custList = query.list();  
String hql = "FROM Customer C WHERE C.age > 25 ORDER BY C.id DESC";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
List<Customer> custList = query.list();

6.> GROUP BY Clause - GROUP BY clause is used with aggregate functions.

String hql = "SELECT COUNT(\*), C.firstName FROM Customer C " + "GROUP BY C.firstName";  
Query query = session.createQuery(hql);  
List<Object[]> objLst = query.list();  
String hql = "SELECT COUNT(\*), C.firstName FROM Customer C " + "GROUP BY C.firstName";  
Query query = session.createQuery(hql);  
List<Object[]> objLst = query.list();

7.> Named Parameters - Named Parameters is used to get inputs from users and helpful to defend against SQL injection attacks.

String hql = "FROM Customer C WHERE C.id = :customerId";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
query.setParameter("customerId",1);  
List<Customer> custList = query.list();  
String hql = "FROM Customer C WHERE C.id = :customerId";  
Query<Customer> query = session.createQuery(hql, Customer.class);  
query.setParameter("customerId",1);  
List<Customer> custList = query.list();

8.> UPDATE Clause - UPDATE clause is used to modify properties of objects.

String hql = "UPDATE Customer set age = :custAge WHERE id = :customerId";  
Query query = session.createQuery(hql);  
query.setParameter("custAge",26);  
query.setParameter("customerId",3);  
int affectedRows = query.executeUpdate();

9.> DELETE Clause - DELETE clause is used to delete objects.

String hql = "DELETE FROM Customer WHERE id = :customerId";  
Query query = session.createQuery(hq0l);  
query.setParameter("customerId", 2);  
int affectedRows = query.executeUpdate();

What is HQL (Hibernate Query Language)?

Hibernate Query Language is known as an object oriented query language. It is like structured query language (SQL).

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

Hibernate offers a query language that embodies a very powerful and flexible mechanism to query, store, update, and retrieve objects from a database. This language, the Hibernate query Language (HQL), is an object-oriented extension to SQL.

@UniqueConstraint and @Column Unique attribute.

@UniqueConstraint and unique attribute of @Column instructs schema/DDL generation tool to generate the corresponding unique constraints however using that attributes on POJO doesn't implement constraints itself.

**@Entity**

**@Table**(uniqueConstraints=

**@UniqueConstraint**(columnNames = {"email", "empl\_id"}))

**public** **class** **EntityClass** {

...

}

**@Entity**

**@Table**(name="USER\_TABLE")

**public** **class** **User**{

**@Id**

**@Column**(name = "ID")

**private** String id;

**@Column**(name = "SSN", unique=**true**)

**private** String ssnumber;

Difference between @NotNull, @NotEmpty and @NotBlank.

**@NotNull**

CharSequence, Collection, Map or Array object cannot be null, however can be empty.

**@NotEmpty**

The CharSequence, Collection, Map or Array object cannot be null and not empty (size > 0).

**@NotBlank**

The string is not null and the length is greater than zero.

**Here are the examples**:

String test = null;

@NotNull: false

@NotEmpty: false

@NotBlank: false

String test = "";

@NotNull: true

@NotEmpty: false

@NotBlank: false

String test = " ";

@NotNull: true

@NotEmpty: true

@NotBlank: false

String name = "Some text";

@NotNull: true

@NotEmpty: true

@NotBlank: true

Different between session.get() and session.load() in Java.

|  |  |
| --- | --- |
| load() | get() |
| Only use the load() method when the object exists in DB | use get() if you are unsure about the existence of object. |
| load() method throws an exception if the unique identifier could not be found. | get() method returns null if the unique identifier not found in the database |
| load() returns a proxy and will not hit database unil the proxy is  invoked to access properties. | get() will hit the database always. |

Difference between save and persist.

Save()

1. Returns generated Id after saving. The return type is serializable.
2. Saves the value to DB immediately and keeps track of the entity until the end of the session(I have tried to change the entity values outside of the transaction, it does not show any effect when session commits)
3. Does not save the changes to the db outside of the transaction.
4. Assigns the generated id to the entity you are persisting
5. Session.save() for a detached object will create a new row in the table.

Persist()

1. Does not returns generated Id after saving. Its void return type.
2. Saves the value to DB immediately and keeps track of the entity until the end of the session.(I have tried to change the entity values outside of the transaction, it does not show any effect when session commits)
3. Does not save the changes to the db outside of the transaction.
4. Assigns the generated id to the entity you are persisting
5. session.persist() for a detached object will throw PersistentObjectException as it is not allowed.

In a Parent-child relationship, how will you only access the columns from parent?

Lazy loading.

Difference between Hibernate createCriteria, createQuery, createSQLQuery.

There are **3**different ways we can create a SQL query in Hibernate.

1. session.createQuery()
2. session.createSQLQuery()
3. session.createCriteria()

**Session.createQuery()**

creates Query object using the HQL syntax. For example:

Query query = session.createQuery("from Student s where s.name like 'k%'");

**Session.createSQLQuery()**

Creates Query object using the native SQL syntax. For example:

Query query = session.createSQLQuery("Select \* from Student");

**Session.createCriteria()**  
Criteria object for setting the query parameters.  
This feature help user to create queries without writing any SQL or HQL.

What are the most common methods of Hibernate configuration?

The most common methods of Hibernate configuration are:

* Programmatic configuration
* XML configuration (using hibernate.cfg.xml)
* Annotations

What is ORM ?

ORM stands for object/relational mapping. ORM is the automated persistence of objects in a Java application to the tables in a relational database.

What is hibernate?

Hibernate framework simplifies the development of java application to interact with the database. Hibernate is an open source, lightweight, ORM (Object Relational Mapping) tool.

What does ORM consists of ?

An ORM solution consists of the following four components:

* API for performing basic CRUD operations
* API to express queries refering to classes
* Facilities to specify metadata
* Optimization facilities : dirty checking,lazy associations fetching

@Repository

annotation that marks or clarifies the specific class as a Data Access Object.

@Embeddable

annotation specifies that the class will be used as a component. This class cannot have a primary key of its own, it uses the enclosing class primary key.

@Service

annotate service layer level classes.

@Controller

annotates presentation layer level classes.

Differentiate JPA and hibernate.

JPA is the specification for an ORM framework whereas hibernate is the implmentation.

What are the ORM levels ?

The ORM levels are,

* Pure relational (stored procedure.)
* Light objects mapping (JDBC)
* Medium object mapping
* Full object Mapping (composition,inheritance, polymorphism, persistence by reachability)

Why do you need ORM tools like hibernate?

* Improved productivity
  + High-level object-oriented API
  + Less Java code to write
  + No SQL to write
* Improved performance
  + Sophisticated caching
  + Lazy loading
  + Eager loading
* Improved maintainability
  + A lot less code to write
* Improved portability
  + ORM framework generates database-specific SQL for you

core interfaces of Hibernate.

The core interfaces of Hibernate framework are:

* Configuration
* SessionFactory
* Session
* Query
* Criteria
* Transaction

Is SessionFactory a thread-safe object?

Yes, SessionFactory is a thread-safe object, many threads can access it simultaneously.

What role does the SessionFactory interface play in Hibernate?

The application obtains Session instances from a SessionFactory. There is typically a single SessionFactory for the whole application created during application initialization. The SessionFactory caches generate SQL statements and other mapping metadata that Hibernate uses at runtime. It also holds cached data that has been read in one unit of work and may be reused in a future unit of work

SessionFactory sessionFactory = configuration.buildSessionFactory();

Is Hibernate Session a thread-safe object?

No, Session is not a thread-safe object, many threads can't access it simultaneously. In other words, you cannot share it between threads.

What are the different states of object in hibernate?

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

* **Transient**: The object is in transient state if it is just created but has no primary key (identifier) and not associated with session.
* **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.
* **Detached**: The object is in detached state if session is closed. After detached state, object comes to persistent state if you call lock() or update() method.

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".

Different types of association in hibernate.

There are four types of association mapping in hibernate.

1. One to One
2. One to Many
3. Many to One
4. Many to Many

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.

What is lazy loading in hibernate?

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.

What is the difference between merge and update ?

Use update() if you are sure that the 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.

How do you define sequence generated primary key in hibernate?

Using <generator> tag.

<id column="USER\_ID" name="id" type="java.lang.Long">

<generator class="sequence">

<param name="table">SEQUENCE\_NAME</param>

<generator>

</id>

What are the Collection types in Hibernate ?

* Bag
* Set
* List
* Array
* Map

What is Hibernate proxy?

The proxy attribute enables lazy initialization of persistent instances of the class. 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.

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.

What are the types of inheritance models in Hibernate?

There are three types of inheritance models in Hibernate:

* Table per class hierarchy
* Table per subclass
* Table per concrete class

Benefits of hibernate over JDBC?

* Hibernate is **database independent** while in case of JDBC, developer has to write database specific queries.
* In case of Hibernate developer **doesn't need to be an expert** of writing complex queries as HQL simplifies query writing process while in case of JDBC, its job of developer to write and tune queries.
* In Hibernate, Criteria Query takes care of Query Tuning. In case of JDBC you need to tune your queries.
* **No need to create any connection pool** in case of Hibernate. You can use c3p0. In case of JDBC you need to write your own connection pool.
* Automatic primary key generation.
* Performance benefits like **Lazy initialization**, Outer join fetching, Batch fetching.

Session Interface.

This is the primary interface used by hibernate applications. The instances of this interface are lightweight and are inexpensive to create and destroy. Hibernate sessions are not thread safe.

SessionFactory Interface.

This is a factory that delivers the session objects to hibernate application. Generally there will be a single SessionFactory for the whole application and it will be shared among all the application threads.

Configuration Interface.

This interface is used to configure and bootstrap hibernate. The instance of this interface is used by the application in order to specify the location of hibernate specific mapping documents.

Transaction Interface.

This is an optional interface but the above three interfaces are mandatory in each and every application. This interface abstracts the code from any kind of transaction implementations such as JDBC transaction, JTA transaction.

Query and Criteria Interface.

This interface allows the user to perform queries and also control the flow of the query execution.

What are Callback interfaces?

These interfaces are used in the application to receive a notification when some object events occur. Like when an object is loaded, saved or deleted. There is no need to implement callbacks in hibernate applications, but they're useful for implementing certain kinds of generic functionality.

What is POJO?

POJO stands for plain old java objects. These are just basic JavaBeans that have defined setter and getter methods for all the properties that are there in that bean. Besides they can also have some business logic related to that property.

Detached Criteria

The detached criteria allows you to create the query without Session. Then you can execute the search in an arbitrary session.

Using a DetachedCriteria is exactly the same as a Criteria except you can do the initial creation and setup of your query without having access to the session. When it comes time to run your query, you must convert it to an executable query with getExecutableCriteria(session).

DetachedCriteria query = DetachedCriteria.forClass(Person.class)

.add( Property.forName("sex").eq('F') );

Transaction txn = session.beginTransaction();

List results = query.getExecutableCriteria(session).setMaxResults(**100**).list();

txn.commit();

session.close();

Advantages of Detached criteria.

* useful to make join conditions, subselects, and to query outside the current session.
* code reuse

Explain N+1 SELECT problem in Hibernate.

Hibernate ends up executing N+1 SQL queries to populate a collection of N elements referred as N+1 SELECT problem that occurs due to N+1 lazy loading and load on demand fetching strategy.

Let's say you have a collection of company objects (database rows), and each company has a collection of employee objects (database rows). In other words, company:employee is a 1-to-many relationship. If you need to iterate through all the companies, and for each one, print out a list of the employees, you have to perform one select to get all companies, and then N additional selects to find the list of employees, where N is the total number of companies.

List the strategies to eliminate the N+1 SELECT problem in Hibernate.

Here are some of the strategies to solve the N+1 problem.

* pre-fetching in batches, this will reduce N+1 problem to N/K + 1 problem where K is size of batch.
* subselect fetching strategy.
* and disable lazy loading.

Explain high level architecture of Hibernate framework.

The important components in Hibernate architecture are,

* SessionFactory,
* Session,
* Persistent objects,
* and Transaction.

org.hibernate.**SessionFactory** : Refers to the cache of compiled mappings (hbm?s or annotation based) for a single database.

org.hibernate. **Session** : Represents a conversation between the java application and the persistent store. It is a wrapper around JDBC java.sql.Connection.

**Persistent objects** : JavaBeans/POJOs with the persistent state and business function associated with one org.hibernate.Session.

org.hibernate.**Transaction** used by Java application to specify atomic units of work. It abstracts the application from the underlying JDBC, JTA or CORBA transaction.

How do we see hibernate generated SQL on console?

Setting the property **show\_sql** to true at the hibernate configuration file enable viewing SQL on the console for debugging purposes.

<property name="show\_sql">true</property>

Hibernate: Can a Entity Class be declared final?

**Yes**, a Hibernate Entity class can be declared final, however it is not a good practice.

Hibernate uses the proxy pattern for performance improvement during lazy association, by making an entity final, Hibernate will no longer be able to use a proxy as Java doesn't allow the final class to be extended.

How to call stored procedure in Hibernate?

In Hibernate, there are 3 approaches to call a database store procedure.

* Use **createSQLQuery()**to call a store procedure directly.
* Use **@NamedNativeQueries** annotation.
* Declare your store procedure inside the **sql-query** tag.

Explain transaction management in hibernate.

In hibernate framework, we have **Transaction** interface that defines the unit of work. It maintains abstraction from the transaction implementation (JTA,JDBC).

A transaction is associated with Session and instantiated by calling **session.beginTransaction**().

The transaction interfaces defines the below methods.

* void **begin**() starts a new transaction.
* void **commit**() ends the unit of work unless we are in FlushMode.NEVER.
* void **rollback**() forces this transaction to rollback.
* void **setTimeout**(int seconds) it sets a transaction timeout for any transaction started by a subsequent call to begin on this instance.
* boolean **isAlive**() checks if the transaction is still alive.
* void **registerSynchronization**(Synchronization s) registers a user synchronization callback for this transaction.
* boolean **wasCommited**() checks if the transaction is commited successfully.
* boolean **wasRolledBack**() checks if the transaction is rolledback successfully.

Difference between fetch mode and fetch type in Hibernate.

FetchType **(Lazy/Eager)**instructs whether the entity to be loaded eagerly or lazy, when there is a call.

FetchMode **(Select/Join)**instructs whether the entitity to be loaded with additional select or in one query with join or subselect.

What is the default fetch mode in Hibernate?

If the Hibernate annotation @Fetch is not present on a field, then the default FetchMode for this field is decided based on the FetchType.

If the field has FetchType = EAGER, then FetchMode = JOIN. Otherwise, FetchMode = SELECT.

What is the difference between FetchType.LAZY and FetchType.EAGER in Java Persistence API?

The EAGER strategy is a requirement on the persistence provider runtime that data must be eagerly fetched. The LAZY strategy is a hint to the persistence provider runtime that data should be fetched lazily when it is first accessed.

Difference between using a @OneToMany and @ElementCollection annotation in Hibernate.

@ElementCollection maps non-entities (embeddable or basic) while @OneToMany is used to map entities.

Explain the features introduced in Hibernate 5.

Support of Java 8 in Hibernate 5 is one of the major changes and it supports Java 8 Date and Time API.

Hibernate 5 supports full-text search through Lucene 5.

Hibernate 5 through OGM provides the support for persistence to the NoSQL Databases such as MongoDB, REDIS.

@NamedQuery and other Hibernate annotations are now repeatable and can be assigned multiple times.This eliminates the use of container annotations such as @NamedQueries.

Which throws Exception when no row found: session.load vs session.get.

session.load() will throw an ObjectNotFoundException when no row found while session.get returns null.

Difference between criteria and HQL In hibernate.

|  |  |
| --- | --- |
| **Criteria.** | **HQL query** |
| Criteria query performs only SELECT operations. | HQL query performs both SELECT and NON-SELECT operations. HQL can be used to perform SELECT, INSERT, UPDATE, DELETE. |
| Criteria supports pagination. | HQL doesn't support pagination. |
| Criteria query is safe from SQL injection because of dynamic query generation. | HQL queries are either fixed or parametrized, SQL injection may happen if the developer doesn't parameterize properly. |

Explain save, persist, saveOrUpdate and other object save methods in Hibernate.

Hibernate provides a handful of methods that store or update the object into the database. They are,

1. save().
2. update().
3. saveOrUpdate().
4. saveOrUpdateCopy().
5. merge().
6. and persist().

The purpose of these methods is as follows.

**save** method persists an entity. It assigns an identifier if entity doesn't exist in the database. If exists, save method performs an update. In both cases save method returns the generated ID of the entity.

**Update** method updates the existing object using identifier. If the identifier does not exist, it throws an exception.

**saveOrUpdate** method checks if the object is transient (i.e. it has no identifier) and if so it performs save that will make it persistent by generating an identifier and assigning it to session. If the object has an identifier already, it performs update().

**saveOrUpdateCopy** method is deprecated and no longer in use. Use merge method instead.

**merge** method is used to update the object to the database in any state of the session.

**persist** method makes a transient object persistent. However, it doesn't guarantee that the identifier value will be assigned to the persistent instance immediately, thats why the method does not return generated ID.

Refreshing Entity objects using refresh() Method in Hibernate.

Sometimes, it is required to re-load an object and its collections at a time when the application database is modified with some external application or database triggers and thus corresponding hibernate entity in your becomes out of sync with its database representation.

In this case, one can use session.refresh() method to re-populate the entity with latest data available in database.

What is the advantage of using session.lock() in hibernate?

session.lock() method is used to reattach an object which has been detached earlier.

Using session.lock() does not check for any data synchronization in database while reattaching the object and hence may lead to lack of synchronization in data.

Difference between sorted and ordered collection in hibernate.

Sorted collection is the way of sorting a collection by leveraging the sorting features provided by the Java collections framework. This sorting uses Java comparator and it takes place in the memory of JVM in which Hibernate is running, once the data being read from database.

Ordered collection refers to the sorting of a collection by specifying the order-by clause when retrieval.

Ordered collection is preferred for larger data set whereas if the collection is considerably small, sorted collection is preferred. Smaller the collection lesser impact on the JVM memory.

How do I map a composite key in Hibernate?

The **EmbeddedId** and **IdClass** annotations are used to denote composite primary keys.

The following are some of the rules that is applied for composite primary keys.

* The primary key class must be public and must have a public no-arg constructor.
* The primary key class must be serializable.
* The primary key class must define equals and hashCode methods.

**Using EmbeddedId.**

**@Embeddable**

**public class** **EntityKey** **implements** Serializable {

**protected** Integer compositeKey1;

**protected** Integer compositeKey2;

**public** **EntityKey**() {}

**public** **EntityKey**(Integer key1, Integer key2) {

**this**.compositeKey1 = key1;

**this**.compositeKey2 = key2;

}

// equals, hashCode

}

**@Entity**

**public** **class** **HibernateEntity** **implements** Serializable {

**@EmbeddedId**

**private** EntityKey primaryKey;

**private** String description;

//...

}

**Using IdClass.**

**public** **class** **EntityKey** **implements** Serializable {

**protected** Integer compositeKey1;

**protected** Integer compositeKey2;

**public** **EntityKey**() {}

**public** **EntityKey**(Integer key1, Integer key2) {

**this**.compositeKey1 = key1;

**this**.compositeKey2 = key2;

}

// equals, hashCode

}

**@Entity**

**@IdClass**(EntityKey.class)

**public** **class** **HibernateEntity** **implements** Serializable {

**@Id**

**private** Integer compositeKey1;

**@Id**

**private** Integer compositeKey2;

**private** String description;

//...

}

List out the design patterns used in Hibernate framework.

* **Domain Model Pattern** - An object model of the domain that incorporates both behavior and data,
* Data Access Object (DAO) Design Pattern,
* Abstract Factory,
* Data Mapper,
* **Proxy** for lazy loading,
* Object-Relational Mapping (ORM),
* **Query Object** for Criterion API,
* and facade.

Why hibernate is preferred to use than JDBC for database interaction in various Java applications?

Hibernate provides an Object oriented view of the database by mapping the various classes to the database tables. This facilitates Object oriented thinking rather than relational and hence increases productivity.

What is difference between openSession and getCurrentSession?

Hibernate SessionFactory getCurrentSession() method returns the session bound to the context. Since this session object belongs to the hibernate context, we don't need to close it. Once the session factory is closed, this session object gets closed.To enable this feature, we need to configure it in hibernate configuration file.

<property name="hibernate.current\_session\_context\_class">thread</property>

Hibernate SessionFactory openSession() method always opens a new session. We should close this session object once we are done with all the database operations. We should open a new session for each request in multi-threaded environment.

Difference between persistence.xml and hibernate.cfg.xml.

JPA leverages metadata from persistence.xml while Hibernate API uses hibernate.cfg.xml.

What is the latest version of Hibernate in use?

The latest version of **hibernate 5.2**released on August 2016.

What are the best practices for defining your Hibernate persistent classes?

The persistent class cannot be final.

You must have a default no-argument constructor for your persistent classes.

getXXX() (getters) and setXXX(mutator/setter) methods for all your persistable instance variables should be available.

You should implement the equals() and hashCode() methods based on your business key.

It is recommended to implement the Serializable interface.

How do I implement Joins in Hibernate?

There are several ways to implement joins in hibernate.

* By using associations such as one-to-one, one-to-many mappings.
* Using JOIN in the HQL query. There is another form 'join fetch' to load associated data simultaneously and it is no lazy loading.
* execute native sql query and use join keyword.

Explain Component mapping in hibernate.

A Component mapping is a mapping for a class having a reference to another class as a member variable.

Hibernate: Can we perform collection mapping with One-to-One and Many-to-One?

No. In Hibernate, collection mapping can only be performed with One-to-Many and Many-to-Many relationship.

How do I create an immutable class in hibernate?

Configuring the entity class property mutable to false (mutable="false"), class becomes an immutable class. By default, it is mutable="true".

Hibernate: Can we switch to different relational database without any code changes?

Yes. In Hibernate we can switch to different database by changing appropriate SQL Dialect configuration.

How do I implement JPA for No-SQL database like mongoDB?

**Hibernate OGM** provides Java Persistence (JPA) support for NoSQL solutions. It reuses Hibernate ORM's engine however it persists entities into a NoSQL datastore like mongoDB rather than a relational database like mySQL, Oracle etc.

What happens when Hibernate Entity bean has no no-args constructor?

Hibernate will fail to instantiate the entity bean and issues HibernateException.

Hibernate uses Reflection API Class.newInstance() to create instance of Entity beans that requires requires no-args constructor, usually when you call get() or load() methods. The instance of the entity bean cannot be created when there is no no-arg (default) constructor.

How do I lookup application server JNDI DataSource using Hibernate?

For web applications, it is always a good practice to allow servlet container to manage the connection pool. This is the reason we define JNDI resource for DataSource and we look it up in the web application. Hibernate need to used the below property to configure and lookup by JNDI DataSource name.

<property name="hibernate.connection.datasource">java:comp/env/jdbc/myTestDB</property>

How do I reattach any detached objects in Hibernate?

Using session.merge() method. Objects that are detached and are no longer associated with any persistent entities can be reattached by calling session.merge() method.

Explain attribute oriented programming in Hibernate.

In Attribute oriented programming, one can add Meta data or attributes in the source code to signify the code. For hibernate, attribute oriented programming is enabled by an engine called XDoclet.

Does hibernate support polymorphism?

Yes, hibernate provides complete support to polymorphism. Polymorphism queries and associations are supported at all the mapping strategies of hibernate.

What are derived properties in hibernate?

Derived properties are those which are not mapped to any columns of a database table. Such properties are calculated at runtime by evaluation of any expressions.

transient annotation is used to denote such properties.

Use of QBC API in Hibernate.

Hibernate Query By Criteria (QBC) API is used to create queries by manipulation of criteria objects at runtime.

What do you mean by ORM metadata in hibernate.

All the mapping between entity class and database table, properties and columns, Java types and SQL types etc is referred as ORM metadata.

Difference between managed and hibernate associations.

Managed associations corresponds to the container managed persistence and it is bi-directional while hibernate associations are unidirectional.

Explain HibernateTemplate.

HibernateTemplate is a helper utility class that provides different methods for querying/retrieving data from the database. It also converts checked Hibernate Exceptions to unchecked DataAccessExceptions.

It manages the session and transactions by automatically opening and closing when you execute the code.

List the four ORM levels in hibernate.

Following are the four ORM levels in hibernate.

* Pure Relational.
* Light Object Mapping.
* Medium Object Mapping.
* and Full Object Mapping.

Explain pure relational ORM.

The entire application along with the user interface (UI) is designed on the basis of relational model and SQL-based relational operations.

Explain light object mapping level of ORM.

The entities are represented as Java classes that are mapped manually to the relational tables. The code is abstracted/hidden from the business logic using specific design patterns.

This approach is desirable for applications with less number of entities, or applications with common, metadata-driven data models.

What is the use of version property in hibernate?

Version property in hibernate helps to identify whether an object is in transient state or in detached state.

Explain medium object mapping in hibernate.

The application is built on the basis of an object model. The SQL code is generated at build time. The associations between objects are supported by the persistence mechanism and queries are specified using an object-oriented expression language.

This is best suited for medium-sized applications with some complex transactions.

Hibernate: What is meant by full object mapping?

Full object mapping supports sophisticated object modeling: composition, inheritance, polymorphism and persistence. The persistence layer implements transparent persistence; persistent classes do not inherit any special base class or have to implement a special interface.

Efficient fetching strategies and caching strategies are implemented transparently to the application.

What do you mean by Extension interfaces in hibernate?

When the built-in functionalities provided by hibernate is not sufficient enough, it provides a way so that user can include other interfaces and implement those interfaces for user desire functionality. These interfaces are referred as Extension interfaces.

How do I persist an image/media in oracle using hibernate ?

Create a database table with column having data type as blob ( or equivalent) and in the HBM configuration file, specify the column type as binary.

What is the role of JMX in hibernate?

JMX API, a standard API manages applications and components in hibernate. JMX provides tools for development of efficient and robust distributed, web based solutions.

How objects can be identified in Hibernate?

Object identification can be done in hibernate in following 3 ways.

* Using Object Identity: Using == operator.
* Using Object Equality: Using equals() method.
* Using database identity: Relational database objects can be identified if they represent same row.

What is the use of version property in hibernate?

Version property is used in hibernate to determine whether an object is in transient state or in detached state.

What is a lazy association in hibernate?

Consider a Parent table associated with a Child table. When we load the Parent table, In lazy association, the Child relationship is loaded when it is needed. This is the default configuration in Hibernate.

What is the use of SchemaValidator in hibernate?

SchemaValidator tool is used to verify if the mapping configured matches the existing database structure.

What are the benefits of detached objects in hibernate?

Detached objects can be passed across layers all the way up to the presentation layer without having to use any DTOs (Data Transfer Objects). You can later on re-attach the detached objects to another session.

What is CRUD?

A CRUD operation deals with creating, retriving , updating and deleting from the table.

What are Scalar queries in Hibernate?

SQL queries that gets a list of scalars (values) is referred as scalar queries.

sess.createSQLQuery("SELECT \* FROM Emp").list();

These will return a List of Object arrays (Object[]) with scalar values for each column in the Emp table.

What are entity query in hibernate?

Entity query gets entity objects from a native sql query using addEntity().

sess.createSQLQuery("SELECT \* FROM Emp").addEntity(Employee.class);

Hibernate: How do you add a criteria to a query?

Session.createCriteria creates a new Criteria instance, for the given entity class, or a superclass of an entity class.

Define persistent classes in hibernate.

Java classes objects or instances that will be stored in database tables are called persistent classes in Hibernate.

Why do we batch processing in hibernate?

Batch processing helps performing data load operations involving high volume transactions that minimize the time required and also is memory efficient.

Hibernate: Is HQL query case sensitive?

Yes. It is.

Explain the role of Configurationclass in Hibernate ?

The org.hibernate.cfg.Configuration is used to build an immutable org.hibernate. SessionFactoryobject . Configuration class object activates hibernate software and configure () is the factory method of hibernate.cfg.Configuration class which reads configuration properties from hibernate.cfg.xml file. builtSessionFactory() method uses hibernate.cfg.xml properties of Configuration object Creates jdbc connection pool.

How to prevent concurrent update in Hibernate?

Using Automatic Versioning Hibernate can perform automatic optimistic concurrency control. It can automatically detect if a concurrent modification occurred during user think time.

Hibernate: disadvantages of using HQL ?

HQL queries cannot perform DDL operations.

HQL queries cannot insert single record into table.

An HQL query gives negligible performance degradation because of conversions when compared to SQL.

HQL queries cannot call PL/SQL program.

Hibernate:advantages of using HQL.

HQL queries are database independent.

HQL queries are object level queries and it returns hibernate pojo class objects as results.

HQL queries and keywords are very much similar to SQL queries.

HQL queries support operators, expressions, conditions ,joins, sub queries, aggregate functions etc.

Explain flush() method in Hibernate.

Flushing the session forces Hibernate to synchronize the in-memory state of the Session with the database (i.e. to write changes to the database). By default, Hibernate will flush changes automatically for you,

* before query executions,
* when a transaction is committed.

Hibernate: Difference commit() vs flush().

flush(): Flushing is the process of synchronizing the underlying persistent store with persistable state held in memory.it will update or insert into your tables in the running transaction, but it may not commit those changes.

Commit(): Commit will make the database commit.When you have a persisted object and you change a value on it, it becomes dirty and hibernate needs to flush these changes to your persistence layer.So You should commit but it also ends the unit of work.

Explain Hibernate named query.

Hibernate named queries lets developer to put all HQL into the XML mapping file or via annotation so that it is easy to maintain and separated from Java code.

The named query is supported in both HQL or native SQL.

The queries can be retrieved using the query names as shown below.

Query query = session.getNamedQuery("HQL\_GET\_ALL\_EMPLOYEE");

What does <![CDATA[]]> in XML mean?

CDATA stands for **Character Data** and it means that the data in between these strings includes data that could be interpreted as XML markup, but should not be.

Explain SQL Dialect in Hibernate.

Hibernate is database agnostic and it can work with different databases. However, databases have native SQL variations, and its own set of SQL standard implementations. Therefore at some point hibernate has to use database specific SQL. Hibernate uses "dialect" configuration to know which database you are using so that it can switch to the database specific SQL generator code wherever/whenever necessary.

What is hibernate proxy?

The proxy attribute facilitates the **lazy initialization** of persistent instances of the class. 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.

Explain hibernate interceptors.

Hibernate interceptor is a powerful feature that allows application to react to certain events that occur inside Hibernate. This allows for the implementation of generic functionality and the extension of Hibernate functionality.

Explain Hibernate Disjunction.

Hibernate Disjunction adds **multiple restrictions/coditions**to the HQL query joined by **"OR"** condition.

What is Hibernate Conjunction?

Hibernate Conjunction adds **multiple restrictions/coditions**to the HQL query joined by **"AND"** condition.

Explain addScalar method in Hibernate.

addScalar method specifies the result of the query to return objects for individual named columns, rather than entities.

Query myQuery = **new** SqlQuery("Select name as nm from employee");

myQuery.addScalar("nm", String);

How hibernate session is related to the JDBC connection?

A hibernate Session is similar to establishing a JDBC connection to the database. When a Session is created in Hibernate, it open up a JDBC connection to the database. When the session is clouded, its close the JDBC connection. Similarly, when transaction is started on hibernate session, it actually start a JDBC transaction.

What is Hibernate Criteria Transformer?

Hibernate Criteria Transformer is an interface that transforms any result of Hibernate Criteria element.

Criteria criteria = session.createCriteria(Employee.class);

criteria.add(Restrictions.eq("name", "Jeff"));

criteria.setResultTransformer(Criteria.DISTINCT\_ROOT\_ENTITY);

How to prevent SQL Injection in hibernate?

Use named parameters in queries to avoid sql injection.

Query query= sessionFactory.getCurrentSession().createQuery("from UserInfo where userName=:userName");

query.setParameter("username", userName);

How do I print a query with parameter values when using Hibernate?

Enable logging for the categories, org.hibernate.SQL and org.hibernate.type.

# logs SQL statements

log4j.logger.org.hibernate.SQL=debug

# Logs JDBC parameters passed to a query

log4j.logger.org.hibernate.type=trace

The property 'org.hibernate.SQL' is equivalent to hibernate.show\_sql=true, and the second prints the bound parameters.

How do we minimize the number of DB write action in Hibernate?

Hibernate provides dirty checking feature that reduces database write times. Dirty checking feature updates only those fields which require a change while keeps others unchanged.

How to configure application server JNDI DataSource with Hibernate framework?

For web applications, it is ideal to have the servlet container manage the connection pool. This is achieved by defining JNDI resource for DataSource and use it in the web application. Use the below property to set the JNDI DataSource name.

<property name="hibernate.connection.datasource">java:comp/env/jdbc/MyDBJNDI</property>

How to implement Joins in Hibernate?

There are different ways to implement joins in Hibernate.

* Using associations such as one-to-one, one-to-many,
* Using JOIN in the HQL query,
* and by using native sql query with join keyword.

Explain the general flow of hibernate.

* Load configuration file and create instance of configuration class.
* Using configuration object, create SessionFactory object.
* From SessionFactory, create session.
* Create HQL query.
* Execute HQL query and get the results.

What is cascade in Hibernate?

Cascading consists in propagating the Parent entity state transition to one or more Child entities, and it can be used for both unidirectional and bidirectional associations.

How to include query hint in Hibernate criteria?

Using Criteria addQueryHint method, we can specify database specific hint.

Criteria **addQueryHint**(String hint)

Difference in positional parameter in JDBC create Query and Hibernate.

In JDBC positional parameter index starts at 1, while it starts at 0 in Hibernate create sql query.

Explain Formula annotation in Hibernate.

@Formula annotation is used to calculate a given entity attribute using an SQL query expression. It defines a formula (derived value) which is a SQL fragment that acts as a @Column alternative in most cases. It represents read-only state.

**@Entity**

**@Table**(name="area")

**public** **class** **Area** **implements** Serializable {

**private** **static** **final** **long** serialVersionUID = **1L**;

**@Id**

**@Column**(name="id")

**private** **int** id;

**@Column**(name="length")

**private** **int** length;

**@Column**(name="width")

**private** **int** width;

**@Formula**(" length \* width ")

**public** **long** area;

...

}

In the example, area property value is calculated using the length and the width properties.

How to limit query results in HQL?

Use Query.setMaxResults() to limit the result.

How do I order the selected entity when using CriteriaQuery?

You can define an ORDER BY clause with the orderBy method of the CriteriaQuery interface and the asc or desc method of the CriteriaBuilder interface.

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Employee> cq = cb.createQuery(Employee.class);

Root<Book> root = cq.from(Employee.class);

cq.orderBy(cb.asc(root.get(Employee\_.firstName)));

// Execute query with pagination

List<Employee> emplList = em.createQuery(cq).getResultList();

Explain JPA Entity Object Life Cycle.

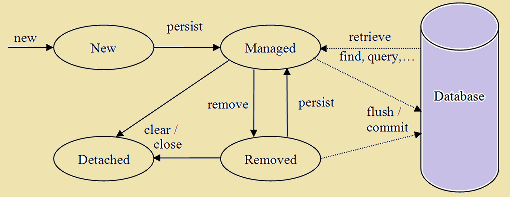
The life cycle of entity objects consists of four states: **New, Managed, Removed and Detached**.

When an entity object is initially created its state is **New**. In this state the object is not yet associated with an EntityManager and has no representation in the database.

An entity object becomes **Managed** when it is persisted to the database via an EntityManagerÂÂÂs persist method, which must be invoked within an active transaction.

A managed entity object can also be retrieved from the database and move it to **removed**, by using the EntityManagerÂÂÂs remove method within an active transaction.

**Detached** represents entity objects that have been disconnected from the EntityManager.



Explain the role of JMX in hibernate.

Java Applications and components are managed in hibernate by a standard API called JMX API. JMX provides tools for development of efficient and robust distributed, web based solutions.

Does hibernate support polymorphism?

Yes, hibernate fully supports polymorphism. Polymorphism queries and polymorphism associations are supported in all mapping strategies of hibernate.

Mention some of the Hibernate Best Practices.

* Identify natural keys for all entities, and map them.
* Place each class mapping in its own file.
* Externalize query strings to make applications more portable.
* Use bind variables to prevent SQL Injection.
* Use hand-coded JDBC sparing: Using SQL defeats the entire purpose of using Hibernate. So, use it sparingly, only when it is performance-critical.
* Prefer lazy fetching for associations. Explicitly disable eager fetching using lazy="false". When join fetching is appropriate to a particular use case, use a query with a left join fetch.
* Use bidirectional associations: In a large application, almost all associations must be navigable in both directions in queries.

Difference between transient and detached objects in hibernate.

**Transient**objects do not have association with the database 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. And of course , garbage collected.

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 is closed.

What happens when a transient mapped object is passed onto a Sessions save?

It has no effect and the transient object retains its state.

Explain cascade and inverse properties in one-many mapping.

There is no relationship between cascade and inverse, both serves different purpose.

What is inverse = true in hibernate mapping?

It defines which side is the parent or the relationship owner for the two entities. Hence, inverse="true" in a Hibernate mapping shows that this designated class is the relationship owner; while the other class is the child.

What is the default fetch type in Hibernate?

The JPA @ManyToOne and @OneToOne annotations are fetched **EAGERly** by default, while the @OneToMany and @ManyToMany relationships are **LAZY** by default.

What is mappedBy attribute in hibernate?

The attribute mappedBy indicates that the entity in this side is the inverse of the relationship, and the owner resides in the "other" entity. This also means that you can access the other table from the class which you've annotated with "mappedBy" (fully bidirectional relationship).

@Entity

**public** **class** **Movie** {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name = "id", updatable = **false**, nullable = **false**)

**private** Long id;

@OneToMany(mappedBy = "movie")

**private** List reviews = **new** ArrayList();

..

}

b = entityManager.find(Movie.class, 122L);

List reviews = b.getReviews();

Assert.assertEquals(b, reviews.get(0).getMovie());

Difference between mappedBy and joinColumn In hibernate.

|  |  |
| --- | --- |
| **mappedBy.** | **joinColumn.** |
| The attribute mappedBy indicates that the entity in this side is the inverse of the relationship, and the owner resides in the "other" entity. | The annotation @JoinColumn indicates that the entity is the owner of the relationship. This means that the corresponding table has a column with a foreign key to the referenced table. |
| You can access the other table from the class which you have annotated with "mappedBy" (fully bidirectional relationship). | You cannot access other table and its one directional. |

### <https://www.edureka.co/blog/interview-questions/java-interview-questions/#hibernate>

### **1. What is Hibernate Framework?**

Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables. Hibernate is Java-based ORM tool that provides a framework for mapping application domain objects to the relational database tables and vice versa.

Hibernate provides a reference implementation of Java Persistence API, that makes it a great choice as ORM tool with benefits of loose coupling. We can use the Hibernate persistence API for CRUD operations. Hibernate framework provide option to map plain old java objects to traditional database tables with the use of JPA annotations as well as XML based configuration.

Similarly, hibernate configurations are flexible and can be done from XML configuration file as well as programmatically.

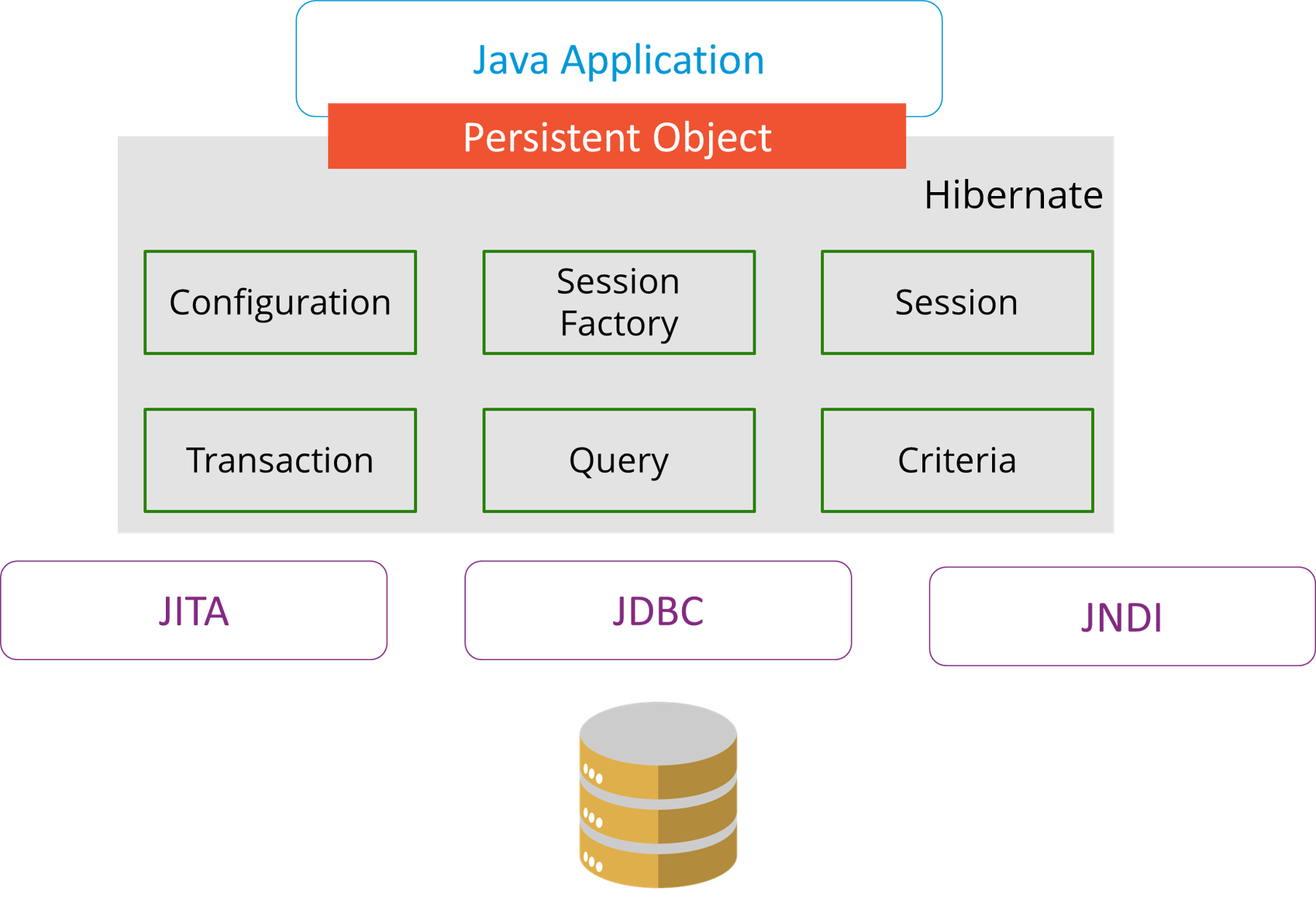
### **2. What are the important benefits of using Hibernate Framework?**

Some of the important benefits of using hibernate framework are:

1. Hibernate eliminates all the boiler-plate code that comes with JDBC and takes care of managing resources, so we can focus on business logic.
2. Hibernate framework provides support for XML as well as JPA annotations, that makes our code implementation independent.
3. Hibernate provides a powerful query language (HQL) that is similar to SQL. However, HQL is fully object-oriented and understands concepts like inheritance, polymorphism, and association.
4. Hibernate is an open source project from Red Hat Community and used worldwide. This makes it a better choice than others because learning curve is small and there are tons of online documentation and help is easily available in forums.
5. Hibernate is easy to integrate with other Java EE frameworks, it’s so popular that Spring Framework provides built-in support for integrating hibernate with Spring applications.
6. Hibernate supports lazy initialization using proxy objects and perform actual database queries only when it’s required.
7. Hibernate cache helps us in getting better performance.
8. For database vendor specific feature, hibernate is suitable because we can also execute native sql queries.

Overall hibernate is the best choice in current market for ORM tool, it contains all the features that you will ever need in an ORM tool.

### **3. Explain Hibernate architecture.**



### **4. What are the differences between get and load methods?**

The differences between get() and load() methods are given below.

|  |  |  |
| --- | --- | --- |
| **No.** | **get()** | **load()** |
| 1) | Returns null if object is not found. | Throws ObjectNotFoundException if an object is not found. |
| 2) | get() method always hit the database. | load() method doesn’t hit the database. |
| 3) | It returns a real object, not a proxy. | It returns a proxy object. |
| 4) | It should be used if you are not sure about the existence of instance. | It should be used if you are sure that the instance exists. |

### **5. What are the advantages of Hibernate over JDBC?**

Some of the important advantages of Hibernate framework over JDBC are:

1. Hibernate removes a lot of boiler-plate code that comes with JDBC API, the code looks cleaner and readable.
2. Hibernate supports inheritance, associations, and collections. These features are not present with JDBC API.
3. Hibernate implicitly provides transaction management, in fact, most of the queries can’t be executed outside transaction. In JDBC API, we need to write code for transaction management using commit and rollback.
4. JDBC API throws SQLException that is a checked exception, so we need to write a lot of try-catch block code. Most of the times it’s redundant in every JDBC call and used for transaction management. Hibernate wraps JDBC exceptions and throw JDBCException or HibernateException un-checked exception, so we don’t need to write code to handle it. Hibernate built-in transaction management removes the usage of try-catch blocks.
5. Hibernate Query Language (HQL) is more object-oriented and close to Java programming language. For JDBC, we need to write native SQL queries.
6. Hibernate supports caching that is better for performance, JDBC queries are not cached hence performance is low.
7. Hibernate provides option through which we can create database tables too, for JDBC tables must exist in the database.
8. Hibernate configuration helps us in using JDBC like connection as well as JNDI DataSource for the connection pool. This is a very important feature in enterprise application and completely missing in JDBC API.
9. Hibernate supports JPA annotations, so the code is independent of the implementation and easily replaceable with other ORM tools. JDBC code is very tightly coupled with the application.

In case you are facing any challenges with these Java interview questions, please comment on your problems in the section below. Apart from this Java Interview Questions Blog, if you want to get trained from professionals on this technology, you can opt for structured training from edureka!

[**Top 50 Hibernate Interview Questions & Answers**](https://career.guru99.com/hibernate-interview-questions/)

**1. What’s Hibernate?**

Hibernate is a popular framework of Java which allows an efficient Object Relational mapping using configuration files in XML format. After java objects mapping to database tables, database is used and handled using Java objects without writing complex database queries.

**2. What is ORM?**  
ORM (Object Relational Mapping) is the fundamental concept of Hibernate framework which maps database tables with Java Objects and then provides various API’s to perform different types of operations on the data tables.

**3. How properties of a class are mapped to the columns of a database table in Hibernate?**

Mappings between class properties and table columns are specified in XML file as in the below example:

**4. What’s the usage of Configuration Interface in hibernate?**

Configuration interface of hibernate framework is used to configure hibernate. It’s also used to bootstrap hibernate. Mapping documents of hibernate are located using this interface.

**5. How can we use new custom interfaces to enhance functionality of built-in interfaces of hibernate?**

We can use extension interfaces in order to add any required functionality which isn’t supported by built-in interfaces.

**6. Should all the mapping files of hibernate have .hbm.xml extension to work properly?**

No, having .hbm.xml extension is a convention and not a requirement for hibernate mapping file names. We can have any extension for these mapping files.

**7. How do we create session factory in hibernate?**

[](https://career.guru99.com/wp-content/uploads/2012/03/hibernate-interview-questions.png)

To create a session factory in hibernate, an object of configuration is created first which refers to the path of configuration file and then for that configuration, session factory is created as given in the example below:

|  |  |
| --- | --- |
| 1  2  3  4 | Configuration config = new Configuration();  config.addResource(&amp;amp;quot;myinstance/configuration.hbm.xml&amp;amp;quot;);  config.setProperties( System.getProperties() );  SessionFactory sessions = config.buildSessionFactory(); |

**8. What are POJOs and what’s their significance?**

POJOs( Plain Old Java Objects) are java beans with proper getter and setter methods for each and every properties.  
Use of POJOs instead of simple java classes results in an efficient and well constructed code.

**9. What’s HQL?**  
HQL is the query language used in Hibernate which is an extension of SQL. HQL is very efficient, simple and flexible query language to do various type of operations on relational database without writing complex database queries.

**10. How can we invoke stored procedures in hibernate?**  
In hibernate we can execute stored procedures using code as below:

[](https://career.guru99.com/wp-content/uploads/2014/07/hibernate_interview_1.png)

**11. What is criteria API?**

Criteria is a simple yet powerful API of hibernate which is used to retrieve entities through criteria object composition.

**12. What are the benefits of using Hibernate template?**  
Following are some key benefits of using Hibernate template:  
a. Session closing is automated.  
b. Interaction with hibernate session is simplified.  
c. Exception handling is automated.

**13. How can we see hibernate generated SQL on console?**  
We need to add following in hibernate configuration file to enable viewing SQL on the console for debugging purposes:

[https://career.guru99.com/wp-content/uploads/2014/07/hibernate_interview_2.png](https://career.guru99.com/wp-content/uploads/2014/07/hibernate_interview_2.png)

**14. What are the two types of collections in hibernate?**  
Following are the two types of collections in hibernate:  
a. Sorted Collection  
b. Order Collection

**15. What’s the difference between session.save() and session.saveOrUpdate() methods in hibernate?**  
Sessionsave() method saves a record only if it’s unique with respect to its primary key and will fail to insert if primary key already exists in the table.  
saveOrUpdate() method inserts a new record if primary key is unique and will update an existing record if primary key exists in the table already.

**16. What the benefits are of hibernate over JDBC?**  
a. Hibernate can be used seamlessly with any type of database as its database independent while in case of JDBC, developer has to write database specific queries.  
b. Using hibernate, developer doesn’t need to be an expert of writing complex queries as HQL simplifies query writing process while in case of JDBC, its job of developer to write and tune queries.  
c. In case of hibernate, there is no need to create connection pools as hibernate does all connection handling automatically while in case of JDBC, connection pools need to be created.

**17. How can we get hibernate statistics?**  
We can get hibernate statistics using getStatistics() method of SessionFactory class as shown below:  
SessionFactory.getStatistics()

**18. What is transient instance state in Hibernate?**  
If an instance is not associated with any persistent context and also, it has never been associated with any persistent context, then it’s said to be in transient state.

**19. How can we reduce database write action times in Hibernate?**  
Hibernate provides dirty checking feature which can be used to reduce database write times. Dirty checking feature of hibernate updates only those fields which require a change while keeps others unchanged.

**20. What’s the usage of callback interfaces in hibernate?**  
Callback interfaces of hibernate are useful in receiving event notifications from objects. For example, when an object is loaded or deleted, an event is generated and notification is sent using callback interfaces.

**21. When an instance goes in detached state in hibernate?**  
When an instance was earlier associated with some persistent context (e.g. a table) and is no longer associated, it’s called to be in detached state.

**22. What the four ORM levels are in hibernate?**  
Following are the four ORM levels in hibernate:  
a. Pure Relational  
b. Light Object Mapping  
c. Medium Object Mapping  
d. Full Object Mapping

**23. What’s transaction management in hibernate? How it works?**  
Transaction management is the process of managing a set of statements or commands. In hibernate; transaction management is done by transaction interface as shown in below code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | Session s = null;  Transaction tr = null;  try {  s = sessionFactory.openSession();  tr = s.beginTransaction();  doTheAction(s);  tr.commit();  } catch (RuntimeException exc) {  tr.rollback();  } finally {  s.close();  } |

**24. What the two methods are of hibernate configuration?**  
We can use any of the following two methods of hibernate configuration:  
a. XML based configuration ( using hibernate.cfg.xml file)  
b. Programmatic configuration ( Using code logic)

**25. What is the default cache service of hibernate?**  
Hibernate supports multiple cache services like EHCache, OSCache, SWARMCache and TreeCache and default cache service of hibernate is EHCache.

**26. What are the two mapping associations used in hibernate?**  
In hibernate; we have following two types of mapping associations between entities:  
a. One-to-One Association  
b. Many-to-Many Association

**27. What’s the usage of Hibernate QBC API?**  
Hibernate Query By Criteria (QBC) API is used to create queries by manipulation of criteria objects at runtime.  
**28. In how many ways, objects can be fetched from database in hibernate?**  
Hibernate provides following four ways to fetch objects from database:  
a. Using HQL  
b. Using identifier  
c. Using Criteria API  
d. Using Standard SQL

**29. How primary key is created by using hibernate?**  
Database primary key is specified in the configuration file hbm.xml. Generator can also be used to specify how primary key is being created in the database.  
In the below example, deptId acts as primary key:

[](https://career.guru99.com/wp-content/uploads/2014/07/hibernate_interview_3.png)

**30. How can we reattach any detached objects in Hibernate?**

Objects which have been detached and are no longer associated with any persistent entities can be reattached by calling session.merge() method of session class.  
**31. What are different ways to disable hibernate second level cache?**

Hibernate second level cache can be disabled using any of the following ways:  
a. By setting use\_second\_level\_cache as false.  
b. By using CACHEMODE.IGNORE  
c. Using cache provider as org.hibernate.cache.NoCacheProvider

**32. What is ORM metadata?**  
All the mapping between classes and tables, properties and columns, Java types and SQL types etc is defined in ORM metadata.

**33. Which one is the default transaction factory in hibernate?**  
With hibernate 3.2, default transaction factory is JDBCTransactionFactory.

**34. What’s the role of JMX in hibernate?**  
Java Applications and components are managed in hibernate by a standard API called JMX API. JMX provides tools for development of efficient and robust distributed, web based solutions.  
**35. How can we bind hibernate session factory to JNDI ?**  
Hibernate session factory can be bound to JNDI by making configuration changes in hibernate.cfg file.

**36. In how many ways objects can be identified in Hibernate?**  
Object identification can be done in hibernate in following three ways:  
a. Using Object Identity: Using == operator.  
b. Using Object Equality: Using equals() method.  
c. Using database identity: Relational database objects can be identified if they represent same row.

**37. What different fetching strategies are of hibernate?**  
Following fetching strategies are available in hibernate:  
1. Join Fetching  
2. Batch Fetching  
3. Select Fetching  
4. Sub-select Fetching  
**38. How mapping of java objects is done with database tables?**  
To map java objects with database tables, we need to have Java beans properties names same as column names of a database table. Then mapping is provided in hbm.xml file as given below:

[](https://career.guru99.com/wp-content/uploads/2014/07/hibernate_interview_4.png)

**39. What are derived properties in hibernate?**  
Derived properties are those properties which are not mapped to any columns of a database table. Such properties are calculated at runtime by evaluation of any expressions.

**40. What is meant by a Named SQL Query in hibernate and how it’s used?**  
Named SQL queries are those queries which are defined in mapping file and are called as required anywhere.  
For example, we can write a SQL query in our XML mapping file as follows:

[](https://career.guru99.com/wp-content/uploads/2014/07/hibernate_interview_5.png)

Then this query can be called as follows:

|  |  |
| --- | --- |
| 1  2  3  4 | List students = session.getNamedQuery(&amp;amp;quot;studentdetails&amp;amp;quot;)  .setString(&amp;amp;quot;TomBrady&amp;amp;quot;, name)  .setMaxResults(50)  .list(); |

**41. What’s the difference between load() and get() method in hibernate?**  
Load() methods results in an exception if the required records isn’t found in the database while get() method returns null when records against the id isn’t found in the database.  
So, ideally we should use Load() method only when we are sure about existence of records against an id.

**42. What’s the use of version property in hibernate?**  
Version property is used in hibernate to know whether an object is in transient state or in detached state.

**43. What is attribute oriented programming?**  
In Attribute oriented programming, a developer can add Meta data (attributes) in the java source code to add more significance in the code. For Java (hibernate), attribute oriented programming is enabled by an engine called XDoclet.

**44. What’s the use of session.lock() in hibernate?**  
session.lock() method of session class is used to reattach an object which has been detached earlier. This method of reattaching doesn’t check for any data synchronization in database while reattaching the object and hence may lead to lack of synchronization in data.

**45. Does hibernate support polymorphism?**  
Yes, hibernate fully supports polymorphism. Polymorphism queries and polymorphism associations are supported in all mapping strategies of hibernate.

**46. What the three inheritance models are of hibernate?**  
Hibernate has following three inheritance models:  
a. Tables Per Concrete Class  
b. Table per class hierarchy  
c. Table per sub-class

**47. How can we map the classes as immutable?**  
If we don’t want an application to update or delete objects of a class in hibernate, we can make the class as immutable by setting mutable=false

**48. What’s general hibernate flow using RDBMS?**  
General hibernate flow involving RDBMS is as follows:  
a. Load configuration file and create object of configuration class.  
b. Using configuration object, create sessionFactory object.  
c. From sessionFactory, get one session.  
d. Create HQL query.  
e. Execute HQL query and get the results. Results will be in the form of a list.

**49. What is Light Object Mapping?**  
Light Object Mapping is one of the levels of ORM quality in which all entities are represented as classes and they are mapped manually.

**50. What’s difference between managed associations and hibernate associations?**  
Managed associations relate to container management persistence and are bi-directional while hibernate associations are unidirectional.