**Hibernate**

**Q-1: What is Hibernate. Explain the feature of Hibernate.**

**A:1-** Hibernate framework simplifies the development of java application to interact with the database.

2-Hibernate is an open source, lightweight, ORM (Object Relational Mapping) tool.

3-Hibernate can be implemented either with XML approach(Hibernate Core) or

4-Application approach(Hibernate Annotation)

5-Hibernate internally follow jdbc procedure to control the DB table.

**There are many advantages of Hibernate Framework. They are as follows:**

1- Hibernate support HQL (Hibernate Query Language),QBC is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries.

2-. Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.

3-Hibernate system is responsible for generating the value required for primary key columns and also support the custom primary key(kclink-001).

4-Hibernate provides many built-in primary key generation algorithms and also supports to implement custom primary key generation algorithms.

1-increment- IncrementGenerator(org.hibernate.id)

2-hilo algorithm

3-uuid algorithm

4-guid

5-identity

6-native

5-Hibernate supports various mapping styles:

1- Simple Mapping

2- Collection Mapping

3-Inheritence Mapping

a- Table per sub class mapping

b- Table per class mapping

c- Table per concrete class mapping

4-Association Mapping

a- One-to-One Mapping

b- One-to-Many Mapping

c- Many-to-Many Mapping

6-Hibernate supports two ways to manage Transactions

a- DriverManager connections

b- DataSource connections

7-Hibernate support two ways to manage Transactions(EJB Transactions)

a- JDBC Transactions

b- JTA Transactions

8- The performance of hibernate framework is fast because cache is internally used in hibernate framework.

Hibernate provides two type of cache(storing the data of various place)

9-Hibernate provides various Query Language

a- HQL(\*)

b- QBC(\*)

c- Native SQL

d- Named SQL(for invoking stored procedure)

**Q-2: What is Java Persistence API (JPA)**

**A:** java Persistence API is a collection of classes and methods to persistently store the vast amounts of data into a database which is provided by the Oracle Corporation. I did not work on JPA.i work on Hibernate

**Q-3: What is ORM?**

**A:** An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.The ORM tool internally uses the JDBC API to interact with the database.

ORM Framwork->

Object - java obj

Relational Entity - tables

Mapping - Realtionship with java obj(Mapping in object oriented

manner)

**Q-4: What are the advantages of Hibernate over JDBC?**

**A:** **1)**    Hibernate is data base independent, your code will work for all ORACLE,MySQL ,SQLServer etc.   
In case of JDBC query must be data base specific. So hibernate based persistance logic is database independent persistance logic and JDBC based persistance logic is database dependent logic.   
**2)**   As Hibernate is set of Objects ,   
**3)** No need to learn SQL language.You can treat TABLE as a Object . Only Java knowledge is need.   
In case of JDBC you need to learn SQL.   
**4)** Dont need Query tuning in case of Hibernate. If you use Criteria Quires in Hibernate then hibernate automatically tuned your query and return best result with performance.   
In case of JDBC you need to tune your queries.   
**5)**  You will get benefit of Cache. Hibernate support two level of cache. First level and 2nd level. So you can store your data into Cache for better performance.   
In case of JDBC you need to implement your java cache .   
**6)** Hibernate supports Query cache and It will provide the statistics about your query and database status.   
JDBC Not provides any statistics.   
**7)** Development fast in case of Hibernate because you dont need to write queries   
**8)**    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   
**9)**  In the xml file you can see all the relations between tables in case of Hibernate. Easy readability.   
**10)**  You can load your objects on start up using lazy=false in case of Hibernate.   
JDBC Dont have such support.   
**11 )** Hibernate Supports automatic versioning of rows but JDBC Not.

**Q-5: What is Hibernate Configuration file .Write the Hibernate Configuration file.**

**A:**

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/subhag</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

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

<property name="hbm2.ddl.auto">**update**</property>

<mapping resource="com/jlcindia/hibernate/Customer.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**Note-1>**

**in the case of XML , Hibernate configuration file we specified**

<mapping resource="com/jlcindia/hibernate/Customer.hbm.xml"/>

</session-factory>

**in the case of Annotation , Hibernate configuration file we specified**

<mapping class="com/jlcindia/hibernate/Student"/>

**Note-2> we can use create instead of update .what is the diff b/w them**

**1-create-(always a new table)**

- if table is not there then it will create

- if table is there then it will drop and create

**2-update-(may be new or existing table)**

- if table is not there then it will create

- if table is there then it will use the existing table

**Q-6: What is the use of hbm2ddl.auto.**

**A:** it will create the table in database automatically

**Q-7: What is the use of Dialect**

**A:** >Hibernate is responsible for generation queries for DB. But SQL queries is not same across the various DB. For this hibernate system contacts the dialect class specified by you and generate the SQL queries which are valid for your DB.

**Q-8: How to create the configuration object up to end of transaction.**

**A:** Configuration cfg = new Configuration();

cfg=cfg.configure();

SessionFactory sf = cfg.buildSessionFactory()

Session session =sf.openSession();

Transaction tx=session.beginTransaction();

Customer cust=new Customer("sri","sri@gm.com",1234,"Banga",25000);

session.save(cust);

tx.commit();

session.close();

**Q-9: How primary key is created by using hibernate?**

**A:** @Id

@GeneratedValue(strategy=GenerationType.AUTO)

**Q-10: Write a simple program to insert the record**

**A:** 1-Customer.java 2-hibernate.cfg.xml 3-Lab2A.java 4-Lab2B.java

1-Customer.java->

import javax.persistence.\*;

@Entity // Annotation

@Table(name="customers") // customer class map with the table customers

public class Customer

{

@Id //ID for PK

@GeneratedValue(strategy=GenerationType.AUTO) // Generated Automatically

@Column(name="cid")

private int cid;

@Column(name="cname")

private String cname;

@Column(name="email")

private String email;

@Column(name="phone")

private long phone;

@Column(name="city")

private String city;

@Column(name="bal")

private double bal;

public Customer()

{

}

public Customer(String cname, String email, long phone, String city, double bal)

{

this.cname = cname;

this.email = email;

this.phone = phone;

this.city = city;

this.bal = bal;

}

//setter and getter method

}

2-hibernate.cfg.xml->

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/subhag</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

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

<mapping class="Customer"/>

</session-factory>

</hibernate-configuration>

4-Lab2A.java->

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

public class Lab2A {

public static void main(String[] args) {

Transaction tx = null;

try

{

AnnotationConfiguration cfg = new AnnotationConfiguration();

cfg=(AnnotationConfiguration)cfg.configure();

SessionFactory sf =cfg.buildSessionFactory()

Session session=sf.openSession();

tx=session.beginTransaction();

Customer cust=new Customer("sri","sri@gm.com",1234,"Banga",25000);

session.save(cust);

tx.commit();

session.close();

System.out.println("Record inserted");

}

catch(Exception e)

{

e.printStackTrace();

if(tx!=null)tx.rollback();

}

}

}

5-Lab2B.java->

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

public class Lab2B {

public static void main(String[] args) {

Transaction tx=null;

try

{

AnnotationConfiguration cfg = new AnnotationConfiguration();

cfg=(AnnotationConfiguration)cfg.configure();

SessionFactory sf =cfg.buildSessionFactory()

Session session=sf.openSession();

Transaction tx=session.beginTransaction();

Customer cust =(Customer)session.load(Customer.class, 1);

System.out.println(cust.getCid()+"\t"+cust.getCname()+"\t"+cust.getEmail+

"\t"+cust.getPhone()+"\t"+cust.getCity()+"\t"+cust.getBal());

tx.commit();

session.close();

}

catch(Exception e)

{

e.printStackTrace();

if(tx!=null)tx.rollback();

}

}

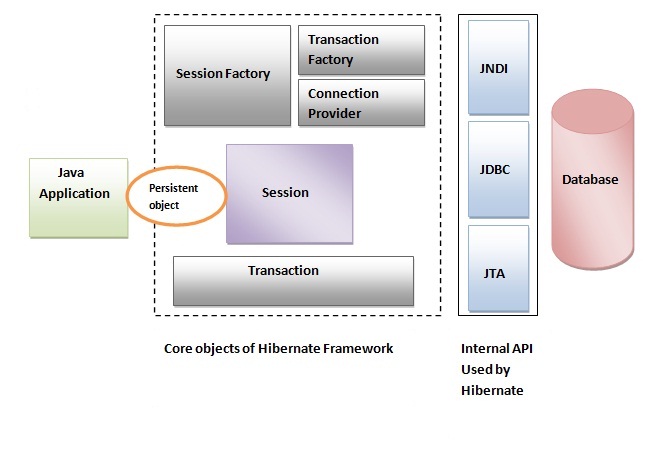
}

**Q-11: Write some important annotations used for Hibernate mapping?**

**A:** Hibernate supports JPA annotations and it has some other annotations in org.hibernate.annotationspackage. Some of the important JPA and hibernate annotations used are:

* 1. **javax.persistence.Entity**: Used with model classes to specify that they are entity beans.
  2. **javax.persistence.Table**: Used with entity beans to define the corresponding table name in database.
  3. **javax.persistence.Access**: Used to define the access type, either field or property. Default value is field and if you want hibernate to use getter/setter methods then you need to set it to property.
  4. **javax.persistence.Id**: Used to define the primary key in the entity bean.
  5. **javax.persistence.EmbeddedId**: Used to define composite primary key in the entity bean.
  6. **javax.persistence.Column**: Used to define the column name in database table.
  7. **javax.persistence.GeneratedValue**: Used to define the strategy to be used for generation of primary key. Used in conjunction with javax.persistence.GenerationType enum.
  8. **javax.persistence.OneToOne**: Used to define the one-to-one mapping between two entity beans. We have other similar annotations as OneToMany, ManyToOne and ManyToMany
  9. **org.hibernate.annotations.Cascade**: Used to define the cascading between two entity beans, used with mappings.

**J javax.persistence.PrimaryKeyJoinColumn**: Used to define the property for foreign key.

**Q-15: Expalin the Hibernate Architecture.**

**Q-11: Write Name some important interfaces of Hibernate framework?**

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

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

**Q-12: Explains the following .**

1. **Configuration or Annotation Configuration**

**2- Configure() method**

**3- Session Factory**

**4- Session**

**5- Transaction**

**A:** All are available in org.hibernate.cfg package.

**1- Configuration or Annotation Configuration->**

1-This is the first class that will be instantiated by Hibernate System.

2- You can use Configuration or AnnotationConfiguration class object for two task

1. Calling configure() method or configure(String) method.
2. Calling bulidsessionFactory() method.

**3-configure() method** is responsible for

1. Identifying configuration document.
2. Reading the data from hibernate configuration document.
3. Identifying all the mapping resources or mapping class.

4-**buildSessionFactory** **method** is responsible for creating SessionFactory object

5-Configuration or Annotation Configuration object is Single Threaded and short lived

6-Once the SessionFactory object is created then there is no use of Configuration or Annotation Configuration object.

**2-Session Factory->**

1-SessionFactory is the factory class used to get the Session objects

2-SessionFactory is responsible to read the hibernate configuration parameters and connect to the database and provide Session objects

3-SessionFactory object is multi threaded and long lived. Internal state of SessionFactory is immutable, so it’s thread safe.Multiple threads can access it simultaneously to get Session instances.

4- SessionFactory also provide methods to get the Class metadata

5- you need to create one SessionFactory per database.

6- When you are using multiple databases then you need to write multiple hibernate configuration document.

**3-Session->**

**1-**Hibernate Session is the interface between java application layer and hibernate

**2-**Session represent the period of time where user can do multiple database operation.

3-Session object is single threaded and short lived.Hibernate Session object is not thread safe,

4- This is the core interface used to perform database operations. Lifecycle of a session is bound by the beginning and end of a transaction.

**4-Transaction->**

1-Transaction is the process of performing multiple database operation as one Atomic unit with all or nothing. Transaction is a single-threaded

2-When transaction is started then following tasks will happen a-> Session cache will be created

b-> Connection will be taken and will be associated with the current session.

3-While transaction is running following tasks will happen

a-> when any object is participated in session operation that will be placed in session cache.

4-When transaction is commited then following tasks will happen

a->pick the object from session cache.

b->prepare the sql query

c->add sql statement to batch

d->send batch to DB

e->Session will be flushed

f->Session cache will be flushed

g->Commit is issue to database

h->Connection will be released.

**Q-13: what is the diff b/w configure() method or configure(String) method.**

A**:configure()>**it will take default hibernate configuration file hibernate.cfg .xml

**configure(String)>**if I will change the name of hibernate configuration file then in this case we have to specified here

**Q-14: Which of the Hibernate Runtime System Objects are Thread – safe.**

**1-Configuration 2- Session Factory 3- Session 4-Transaction**

**A:** 1- Configuration ThreadSafe

2- Session Factory NotThreadSafe

3- Session ThreadSafe

4- Transaction ThreadSafe

**Q-15: What is difference between openSession() and getCurrentSession**

**A: openSession()->1-**It will return a new session object on every call,

2- We can use this method when we decided to manage the Session our self.

3-It does not try to store or pull the session from the current context.

4-If we use this method, we need to **flush() and close()** the session. It does not flush and close() automatically.

**getCurrentSession()->1-** if u want to work with already existing session go for Session.getCurrentSession()

2-This creates a brand new session if one does not exist or uses an existing one if one already exists.

3-It automatically configured with both auto-flush and auto-close attributes as true means Session will be automatically flushed and closed.

4- We can use getCurrentSession() method when our transaction runs long time.  To use this method we need to configure one property in **hibernate.cfg.xml**file

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

**Q-16:What is difference between session.get() and session.load() method?**

**A:** Both are from Session interface, and we will call them as session.get() & session.load().Both will be use for retrieving the object (a row) from the database

**load()** method throws exception when primary key not found.

Only use the load() method if you are sure that the object exists.

**get()** method returns the null value when primary key not found

If you are not sure that the object exists, then use one of the get() methods.

**Q-17: What is difference between session.save(), session.update() methods?**

**Q-18:** **Difference between session.update() and session.lock() in Hibernate ?**

**Q-19: What is difference between session.save() and session. persist() methods?**

**Q-20: Explain the following things.**

**1-save() method->**it is insert the newly created object in databse and generate the identifier value(Basically identifier value will be 0 for this) Like I create a new customer and call save operation,it will store in database and generate the identifier.

**2-update() method->**it is used to update the already persistent object in database and generate the identifier value for this(Basically identifier value will be some non zero for this).Like I load a new customer and call update operation after update of some filed value.it will update it in database.

**3-persist() method->**it is same as save() method but the diff b/w save() and persist method is**Return type of save() method is java.io.**[**Serializable**](http://docs.oracle.com/javase/6/docs/api/java/io/Serializable.html)**.** while **Return type of persist() method is void**(it does not return any value)

**4-saveorUpdate() method->**saveOrUpdate can either INSERT or UPDATE based upon existence of record. saveOrUpdate will check for existence, if record exists it will be updated else a new record will be inserted.

**5-lock() method()->**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.

**Q-21: What is the difference between session.merge() and session.update() method?**

**A:**update method will be applied to object when it is in session i.e if session.close() applies on object then that object is in detached state then if you call update method then it will give error.if you want to update previous object then open the new session object again and with the new session object merge the previous object . **session2.merge(s1);**now into s2 object s1 changes will be merged and saved into database.

Ex>

1-SessionFactory factory = cfg.buildSessionFactory();

**2-Session session1 = factory.openSession();**

3-Student s1 = null;

4-Object o = session1.get(Student.class, new Integer(101));

5-s1 = (Student)o;

6-session1.close();

7-s1.setMarks(97);

**8-Session session2 = factory.openSession();**

9-Student s2 = null;

10-Object o1 = session2.get(Student.class, new Integer(101));

11-s2 = (Student)o1;

12-Transaction tx=session2.beginTransaction();

13-session2.merge(s1);

Explaination->

* See from line numbers 3-6, we just loaded one object s1 into session1 cache and closed session1 at line number 6, so now object s1 in the session1 cache will bedestroyed as session1 cache will expires when ever we say session1.close()
* Now s1 object will be in some RAM location, not in the session1 cache
* here s1 is in detached state, and at line number 7 we modified that detached object s1, now if we call update() method then hibernate will throws an error, because we can update the object in the session only
* So we opened another session [session2] at line number 8,  and again loaded the same student object from the database, but with name s2
* so in this session2, we called **session2.merge(s1)**; now into s2 object s1 changes will be merged and saved into the database

**Q-22: What are the Loading strategies supported by Hibernate.**

**A:**Hibernate supports two Loading strategies.

1. Aggresive Loading or Eager Loading
2. Lazy Loading

**Consider the following Requirement->**

public class Student

{

private int sid;

private String sname;

private String dob;

private String qualification;

**//collection mapping**

private List<String> emails;

private Set<Long>phones;

private Map<String,Long>refs;

**//Association mapping**

private Address address;

private Set<Course>cources;

…………

}

**Note:**

1-Student table is the master table which has the primary key called sid.

2-Other student information related to collection mapping and association mapping will be placed in other 5 table called emails,phones

,refs,address,courses

**Consider that you are loading the student whose sid is 101 as follows**

Student stu=(Student)session.load(Student.class,1);

**--->In the case of Aggressive Loading ,**when you load the persistent object then all of its Collections and Association persisted in other table also will be loaded at the time of loadingthe main Persistent object.

**--->In the case of Lazy Loading ,**when you load the persistent object then all of its Collections and Associations persisted in other table also will not be loaded at the time of loadingthe main Persistent object,will be loaded lazily I.e. whenever you try to access them.

**Q-23: What is the default Loading strategies used.**

**A:** Lazy Loading

**Q-24: Which Loading strategy is best for better Performance.**

**A:** Lazy Loading

**Q-25: can I contact multiple database in hibernate.**

**A:** yes but for this you have multiple configuration file(per database)

**Q-26: What are the lock type available.**

**A:** There aretwotype of Lock available in Hibernate

**1-Premistic Locking->**if this locking applied on column or row then that column or row will not be accessed by any other concurrency running transaction for read-only or for update operation

**2-Optimistic Locking->** if this locking applied on column or row then that column or row will not be accessed by any other concurrency running transaction for read-only but not for update operation

**Q:Which lock style supported/recomandaed in Hibernate**

**A:**optimistic locking

**Q-27: What will happen if we don’t have no-args constructor in Entity bean?**

**A:**

**Q-28:How many mapping available in Hibernate.**

**A:** Hibernate supports various mapping styles:

1- Simple Mapping

2- Collection Mapping

3- Inheritence Mapping

a- Table per sub class mapping

b- Table per class mapping

c- Table per concrete class mapping

4-Association Mapping

a- One-to-One Mapping

b- One-to-Many Mapping

c- Many-to-Many Mapping

**Q-29: What is Simple Mapping**

**A:** when you map your Hibernate Persistence class with simple data types like String,Primitives,Wrappers,Date etc with the corresponding table columns then it is called as simple mapping.

1-you need to use following Annotation for Persistence class(for class)

@Entity

@Table(name="customers")

class Customer

{

....

}

2-you need to use the following Annotations for primary key field

@Id

@Column(name="cid")

private int cid;

3-you need to use the following Annotations for other simple field(for other attribute)

@Column(name="cname")

private String cname;

**Q-30: What is Collection Mapping.**

**A:** when you map your Hibernate Persistence class with collection data type like array,List, Set,Map then you have to use Collection mapping.

A-Persistence Class

@Entity

@Table(name="studnets")

public class Student

{

@Id

@GeneratedValue(strategy=GenerationType.AUTO) //PK Auto\_generated

@Column(name="sid")

private int sid; //PK

@Column(name="sname")

private String sname;

@Column(name="dob")

private String dob;

@Column(name="quali")

private String qualification;

@CollectionOfElements

@JoinTable(name="courses",joinColumns=@JoinColumn(name="sid"))--

>here sid is FK

@IndexColumn(name="idx")

@Column(name="cname")

private String []courses;

@CollectionOfElements

@JoinTable(name="emails",joinColumns=@JoinColumn(name="sid"))

@IndexColumn(name="idx")

@Column(name="emailid")

private List<String> emails;

@CollectionOfElements

@JoinTable(name="marks",joinColumns=@JoinColumn(name="sid"))

@Column(name="marks")

private List<Integer> marks; -->

@CollectionOfElements

@JoinTable(name="phones",joinColumns=@JoinColumn(name="sid"))

@Column(name="phoneNo")

private Set<Long> phones; --> for set we also no need of index

@CollectionOfElements

@JoinTable(name="refs",joinColumns=@JoinColumn(name="sid"))

@Column(name="rphone")

private Map<String,Long> refs;

.......

}

note->

1-@Entity -->for class name we use @Entity

2-@CollectionOfElements -->for any type of collection like array,list,map,set we use this tag

3-@JoinTable(name="courses",joinColumns=@JoinColumn(name="sid"))-->join column on the basis of FK(sid)

4-@IndexColumn(name="idx")-->index column -

5-@Column(name="cname") -->element column

**Q-31: What is Inheritance Mapping. Explain the type also and write the code also of “Table per Sub class” Mapping**

**A:** 1-when multiple Hibernate Persistence class are in inheritence relaionship then

use Inheritence Mapping.

2-you can inplement Inheritence Mapping in 3 ways

-->Table per Sub class mapping (it is best among all)

-->Table per class mapping

-->Table per Concrete class mapping

3-Consider the following Hibernate Persistence class with inheritence relaionship

Student(Main Superclass) Master Table

|

---------------------------

| |

Current Student Old Student

|

----------------------------

| |

Weekday Student Weekend Student

note->for the student information placed in 6 tables its very difficult

to insert data in all 6 tables.

**1-Table per Sub class Mapping->**

**------------------------------------------**

1- in this mapping you need to take one table per one sub class

2- Student is the main super class which will have the master table called mystudents.

3- Every sub class will have its own table.

4-when you save Student class object then

-->only one record will be inserted in mystudents.

5-when you save CurrentStudent class object then

-->one record will be inserted in mystudents.

-->one record will be inserted in cstudents.

6-when you save OldStudent class object then

-->one record will be inserted in mystudents.

-->one record will be inserted in ostudents.

7-when you save WeekdayStudent class object then

-->one record will be inserted in mystudents.

-->one record will be inserted in cstudents.

-->one record will be inserted in wdstudents.

8-when you save WeekendStudent class object then

-->one record will be inserted in mystudents.

-->one record will be inserted in cstudents.

-->one record will be inserted in westudents.

**A-Table Required->**

**1-mystudents**

+-----+-------+---------+--------+----------+

| sid | sname | city | status | totalfee |

+-----+-------+---------+--------+----------+

| | | | | |

| | | | | |

+-----+-------+---------+--------+----------+

**2-cstudents**

+-----+--------+---------+-----------+

| sid | feebal | timings | branch |

+-----+--------+---------+-----------+

| | | | |

| | | | |

+-----+--------+---------+-----------+

**3-ostudents**

+-----+----------+-----------+------+

| sid | ocompany | ocemail | octc |

+-----+----------+-----------+------+

| | | | |

+-----+----------+-----------+------+

**4-wdstudents**

+-----+---------------+------------+------+

| sid | qualification | persentage | yoe |

+-----+---------------+------------+------+

| | | | |

+-----+---------------+------------+------+

**5-westudents**

+-----+----------+-----------+------+

| sid | wcompany | wcemail | wctc |

+-----+----------+-----------+------+

| | | | |

+-----+----------+-----------+------+

**1-Student.java->Super class**

@Entity

@Table(name="mystudents")

@Inheritence(strategy=InheritanceType.JOINED) //we use this things in super

Class

public class Student {

................

}

**2-CurrentStudent.java->**

@Entity

@Table(name="cstudents")

@PrimaryKeyJoinColumn(name="sid") // this is common for every sub class

// join this on the basis of primary key

public class CurrentStudent extends Student {

..................

}

**3-OldStudent.java->**

@Entity

@Table(name="ostudents")

@PrimaryKeyJoinColumn(name="sid")

public class OldStudent extends Student{

..................

}

**4-WeekdayStudent.java->**

@Entity

@Table(name="wdstudents")

@PrimaryKeyJoinColumn(name="sid")

public class WeekdayStudent extends CurrentStudent {

private String qualification;

private String persentage;

private int yoe;

.............

}

**5-WeekendStudent.java->**

@Entity

@Table(name="westudents")

@PrimaryKeyJoinColumn(name="sid")

public class WeekendStudent extends CurrentStudent{

private String wcompany;

private String wcemail;

private double wctc;

.................

}

note->

------

in the case of Table per sub class mapping

--> use the following for super class

@Inheritence(strategy=InheritanceType.JOINED)

--> use the following for all the sub classes

@PrimaryKeyJoinColumn(name="sid")

in Annotation,in configuration document,we need to specify all the classes

in hibernate.cfg.xml in such a way-

<mapping class="Student"/>

<mapping class="CurrentStudent"/>

<mapping class="OldStudent"/>

<mapping class="WeekdayStudent"/>

<mapping class="WeekendStudent"/>

Note:No need to discuss remaning inheritence type.because they are not useful

**2-Table per class Mapping->**

**-------------------------------------**

1-in this mapping you need to take only one table for all the super and sub class.

2-it is also called as Single Table Mapping.(all data reside in single table)

3-in this no sub class required

4-it will show some extra column that contains null value.so it is not good

5-no foreign key required due to only 1 table is here

A-Table required

----------------

->**jlcstudents(total 18 column)**

-----------------------------------------------------------------------------------------

sid | stutype| sname | city | status | totalfee | feebal | timings | branch | ocompany |

----|--------|-------|-------|--------|----------|--------|---------|--------|------------

| | | | | | | | | |

| | | | | | | | | |

-----------------------------------------------------------------------------------------

------------------------------------------------------------------------------------

ocemail| occtc | qualification | percentage | yoe | wcompany | wcemail | wcctc |

-------|--------|---------------|------------|-------|----------|---------|---------|

| | | | | | | |

| | | | | | | |

-------------------------------------------------------------------------------------

**3-Table per Concrete class mapping->**

**----------------------------------------------**

1- in this mapping,you need to take one table for one concrete class

2- when you save Student class object then only one record will be

inserted in mystudents1.

3- when you save CurrentStudent class object then only one record will

be inserted in cstudents1.

4- when you save OldStudent class object then only one record will

be inserted in ostudents1.

5- when you save WeekdayStudent class object then only one record will

be inserted in wdstudents1.

6- when you save WeekendStudent class object then only one record will

be inserted in westudents1.

A-Tables required->

**mystudents1 (5 cols)**

+-----+-------+-------+---------+----------+

| sid | sname | city | status | totalfee |

+-----+-------+-------+---------+----------+

| 1 | sri | Blore | Enabled | 15000 |

+-----+-------+-------+---------+----------+

**cstudents1 (8 cols)**

+-----+-------+-------+---------+----------+--------+---------+-----------+

| sid | sname | city | status | totalfee | feebal | timings | branch |

+-----+-------+-------+---------+----------+--------+---------+-----------+

| 1 | vas | Blore | Enabled | 15000 | 2000 | 6.30PM | Mathikere |

+-----+-------+-------+---------+----------+--------+---------+-----------+

**oldstudents1 (8 cols)**

+-----+-------+-------+---------+----------+----------+-----------+------+

| sid | sname | city | status | totalfee | ocompany | ocemail | octc |

+-----+-------+-------+---------+----------+----------+-----------+------+

| 1 | aa | Blore | Enabled | 15000 | SDSOFT | aa@sd.com | 9 |

+-----+-------+-------+---------+----------+----------+-----------+------+

**wdtudents1 (11 cols)**

+-----+-------+-------+---------+----------+--------+---------+-----------+---------------+------------+------+

| sid | sname | city | status | totalfee | feebal | timings | branch | qualification | persentage | yoe |

+-----+-------+-------+---------+----------+--------+---------+-----------+---------------+------------+------

| 1 | bb | Blore | Enabled | 15000 | 2000 | 6.30PM | Mathikere | M.Sc | 85.5 | 3 |

+-----+-------+-------+---------+----------+--------+---------+-----------+---------------+------------+------

westudents1 (11 cols)

+-----+-------+-------+---------+----------+--------+---------+--------+----------+-----------+------+

| sid | sname | city | status | totalfee | feebal | timings | branch | wcompany | wcemail | wctc |

+-----+-------+-------+---------+----------+--------+---------+--------+----------+-----------+------+

| 1 | cc | Blore | Enabled | 15000 | 2000 | 6.30PM | HSR | SDSOFT | cc@sd.com | 9 |

+-----+-------+-------+---------+----------+--------+---------+--------+----------+-----------+------+

**Q-32: Which Inheritence mapping is best and how.**

**A:** Table per sub class mapping

**Q-33: What is association Mapping.**

**A:** 1- if you want to establish the relationship among different persistence class,then you have to use Association Mapping

2- Depending on the cardinality,there are 3 types of Association Mapping

1-- One to One Mapping

2-- One to Many Mapping

3-- Many to Many Mapping

3- cardinality represents number of objects like(Stud {obj},Add{obj}) participating in Relationship on both the sides.

4-Directionality represents whether we can access the data in one direction only or both the directions.

Ex:

**1- One to One Bi-Directional**

-- One Customer can have One Address and One Address belongs to One Customer.

**2- One to Many Bi-Directional**

-- One Customer can place Many Orders and One Order placed by One Customer.

**3- Many to Many Bi-Directional**

-- One Customer can place Many Accounts and One Account belongs to Many Customers.

-- One Student can join for Many Courses and One Course can belongs to Many Students.

**Q-34:Write the Syntax of-> 1- One to One Mapping\_\_\_Bidirectional**

**2- One to Many ….. \_\_\_Bidirectional**

**3- Many to Many … . \_\_\_Bidirectional**

**A:**

1. **One to One Mapping\_Bidirectional->**

Consider the relationship between Customer and Address

-- One Customer contains One Address

-- One Address belongs to One Customer

Accessing data in reverse direction

Access

Customer class ---------------------------> Address class

| |

| possible |

<-------------------------------------------|

(because of bidirectional mapping)

**A-Table Required->**

**mycustomers (cid is PK)**

+-----+-----------+----------+---------------+-------+-------+

| cid | firstname | lastname | email | phone | aid | here aid is FK

+-----+-----------+----------+---------------+-------+-------+

| 1 | sri | nivas | sri@gmail.com | 1234 | 1 |

+-----+-----------+----------+---------------+-------+-------+

**addresses**

+-----+----------+-------+----------+

| aid | street | city | state | here aid is PK

+-----+----------+-------+----------+ (it is taking Fk of mycustomers)

| 1 | HMT Main | Blore | Karnatka |

+-----+----------+-------+----------+

**Hibernate Persistence class->**

**1-Customer.java**

@Entity

@Table(name=”mycustomers”)

Public class Customer

{

……….

**@OneToOne**

**@joinColumn(name=”aid”)**

Address address;

………..

}

**2-Address.java**

@Entity

@Table(name=”addresses”)

class Address

**{**

**………..**

**@OneToOne**

**@joinColumn(name=”aid”)**

Customer customer;

………..

}

**2-One to Many Mapping\_Bidirectional->**

Consider the relationship between Customer and Request

-- One Customer can place Many Requests

-- One Request belongs to One Customer only

A-Tables Required->

**jlccustomers;**

+-----+-----------+----------+---------------+-------+

| cid | firstname | lastname | email | phone |

+-----+-----------+----------+---------------+-------+

| 1 | sri | nivas | sri@gmail.com | 1234 |

+-----+-----------+----------+---------------+-------+

**requests;**

+-------+------------+-------------+--------+------+

| reqid | reqDate | description | status | cid | cid is Fk

+-------+------------+-------------+--------+------+

| 1 | 22/07/2014 | hello1 | ok1 | 1 |

| 2 | 23/07/2014 | hello2 | ok2 | 1 |

| 3 | 24/07/2014 | hello3 | ok3 | 1 |

+-------+------------+-------------+--------+------+

**Hibernate Persistence class->**

**1-Customer.java**

@Entity

@Table(name=”customers”)

Public class Customer

{

……….

**@OneToMany(mappedBy = “customer”)**

**Private Collection<Request>requests;**

………..

}

**2-Request.java**

**@Entity**

**@Table(name=”requests”)**

**class Request**

**{**

**………..**

**@ManyToOne**

**@joinColumn(name=”cid”,referencedColumnName=”cid”)**

**Customer customer;**

………..

}

**3-Many to Many\_\_Bidirectional->**

Consider the relationship between Customer and Account

-- One Customer can have many Accounts

-- One Account Belongs to Many Customers.

Consider the relationship between Student and Course

-- One Student can join for many Courses

-- One Course can have Many Students

A-Tables Required->

------------------

**jlcstudents;**

+-----+--------+------------------+-------+

| sid | sname | email | phone |(here sid is PK)

+-----+--------+------------------+-------+

| 1 | sri | sri@gmail.com | 9999 |

| 2 | nivas | nivas@gmail.com | 8888 |

| 3 | manish | manish@gmail.com | 7777 |

+-----+--------+------------------+-------+

**jlccourses;**

+-----+-------+----------+------+

| cid | cname | duration | cost |(here cid is PK)

+-----+-------+----------+------+

| 1 | JAVA | 9 | 13 |

| 2 | JSP | 8 | 12 |

| 3 | EJB | 7 | 11 |

+-----+-------+----------+------+

**stu\_cou;(this is the mapping table)**

+-----+-----+

| sid | cid | in this (sid,cid is FK)

+-----+-----+

| 1 | 1 | "sri" student belongs to three course

| 1 | 2 | "java" course belongs to two students

| 1 | 3 |

| 2 | 1 |

| 2 | 2 |

| 3 | 2 |

| 3 | 3 |

+-----+-----+

**Hibernate Persistence class->**

**1-Student.java**

@Entity

@Table(name=”students”)

Public class Student

{

……….

**@ManyToMany**

**@joinTable(name=”stu\_cou”,**

**joinColumns=**

**@joinColumn(name=”sid”,referenceColumnName=”sid”))**

**inversejoinColumns=**

**@joinColumn(name=”cid”,referenceColumnName=”cid”))**

**Private Set<Course> courses;**

………..

}

**2-Course.java**

**@Entity**

**@Table(name=”courses”)**

**class Course**

**{**

**………..**

**@ManyToMany**

**@joinColumn(mappedBy=”courses”)**

**Private Set<Student> students;**

………..

}

**Q-35:How to implement relationships in hibernate?**

**A:** By mapping we can implement the relationship.

**Q-36: What is HQL and what are it’s benefits**

**A:** HQL is an **object-oriented query language**, similar to SQL, but instead of operating on tables and columns, HQL works with persistent objects and their properties.  So it is database independent query language.

**Q-35: Write the HQL Queries for the following statement**

**A:**

1. **Display all the student**

String hql = “from student stu”;

Query q = session.createQuery(hql);

List<Student> list=q.list();

**2-Display all the customer staying in Blore**

String hql = “from Customet c where c.city=?”;

Query q = session.createQuery(hql);

q.setString(0,”Blore”);

List<Customer> list=q.list();

for(Customer c : list)

{

System.out.println(c);

}

**3-Display all the customer staying in Blore with bal>10000**

String hql = “from Customet c where c.city=? and c.cardBal>10000” ;

Query q = session.createQuery(hql);

q.setParameter(0,”Blore”);

q.setParameter(1,10000);

List<Customer> list=q.list();

**Q-36: What is QBC and what are it’s benefits?**

**A:** Hibernate Query By Criteria (QBC) API is used to create queries by manipulation of cieteria object at run time.

**Q-37: Write the QBC Queries for the following statement**

**A:**

**1-Display all the student**

Crieteria ct=session.createCriteria(Student.class);

List<Student> list=ct.list();

for(Student st : list)

{

System.out.println(st);

}

**2-Display all the customer staying in Blore with bal>10000**

Crieteria ct=session.createCriteria(Customer.class);

Ct.add(Expression.eq(“city”,”pune”))

List list=ct.list();

for(Customer c : list)

{

System.out.println(c);

}

**3-Display all the customer staying in Blore with bal>10000**

Crieteria ct=session.createCriteria(Customer.class);

Ct.add(Expression.and(Expression. eq(“city”,”pune”),Expression.gt(“bal”,10000)))

List list=ct.list();

**Q-38:What is criteria API?What is the benefit of Hibernate Criteria API?**

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

**Q-39: What is pagination? Give example through HQL and QBC.**

**A:**if the Hibernate fetches large amount of data(records) from the database, It consumes lot of memory. To consume less RAM memory, the data can Be obtained in installment from the database. So solution is **Pagination.**The idea behind pagination is to divide the large result set into a number of pages and fetching one page at a time. we can programmatically declare how many records should contain each page and from what record.

**BY HQL->**

Query q=session.createQuery(“from Student”);

q.setFirstResult(0);

q.setMaxResult(2);

result = q.list();

**BY QBC->**

Criteria crit = session.createCrieteria(Product.class);

crit.setFirstResult(0);

crit.setMaxResult(4);

**Q-40: How can we see hibernate generated SQL on console?**

**A:** <property name="show\_sql">true</property>

**Q-41: What are built- in primary keys. How to implement own primary key.**

**A:**when you use single column as primary key then it is called as simple primary. use <id>tag or @Id annotation to configure the primary key

**Ex:**Build in primary key algorithm->increment,hilo,sequence,uuid,guid..etc.

**Custom Primary Key ID Generator->(JLC-001)(By using xml)**

1->Write your generator class(SIDGenerator) by implementing IdentifierGenerator interface which is in org.hibernate.id package

2-> override the generate() method as follows

1. Public Serializable generate(SessionImplementor si,Object obj) throws HibernateException

3**->** Write the required id generation logic in generate() method

4-> Register Custom ID Generator in hibernate mapping document

<id column = “sid” name=”sid” type=”string”>

<generator class=”com.jlc.id.SIDGenerator”>

**Q-42: What are composite primary key. How to implement composite primary key.**

**A:**whenyou use combination of two or more column as primary key then it is called as primary key.

Use @Embedded and @Embeddable to configure composite primary key.

**Steps to configure the Composite Primary Key:**

**1->**Write custom Composite Primary Key class by implementing Serializable interface and declare the required fields.mark the class with @Embeddable

Ex:

@Embeddable

public class SID implements Serializable

{

private String bid;

private String sid;

……….

}

**2->**Declare the primary key field type as Composite Primary Key class.

Ex:

@Entity

public class Student

{

@Id

@Embedded

@AttributeOverrides(

{

@AttributeOverride(name="bid",column=@Column(name="bid")),

@AttributeOverride(name="sid",column=@Column(name="sid"))

}

)

private SID studentid;

@Column(name="sname")

private String sname;

@Column(name="email")

private String email;

@Column(name="phone")

private String phone;

public Student(){}

public Student(SID studentid, String sname, String email, String phone) {

super();

this.studentid = studentid;

this.sname = sname;

this.email = email;

this.phone = phone;

}

…………………….

}

**Q-43:Explain Transaction management in Hibernate?**

**A:** Transaction is the process of performing multiple database operation as one Atomic unit with all or nothing.

When all the database operation in the unit are successful the transaction is successful and should be commited and if any one operation is failed then whole transaction should be rollback.

To implement the transaction successfully,you follow the ACID Properties.

**1-Atomicity->**All the operation should be successfully completed then it will Commit transaction otherwise rollback the transaction.

**2-Consistency->**When you are running the transaction you may enforce some Business rules as per the application requirement.your application should be consistent(not produce wrong result)when any business Rule is failed

**3-Isolation->**you can run multiple transaction concurren.These concurrently Running multiple txs should not disturb other transaction.

**4-Durability->**your enterprise data should be available for long time as long As your enterprise application is running.

You have to protect your enterprise from crashes and failure.

**Q-44: What are the Transactional Concurrency Problems.**

**A:**when multiple transaction running concurrently and using a single column or row or table which may cause problems.

The problems coming when multiple transaction running concurrently are called Transactional Concurrency Problems.

**These are the Transaction Concurrency Problems.**

1. Dirty Read Problem
2. Repetable Read Problem
3. Phantom Read Problem

**To avoid these Transaction Concurrency Problems,you have to apply one of the following required Transactional Isolation Levels.**

1-READ\_UNCOMMITED 1 (No Lock)

2-READ\_COMMITED 2

3-REPEATABLE\_READ 4

4- SERIALIZABLE 8 (this is constant serilization not IO serilization)

**1-Dirty Read Problem->**1**-**when a Transaction reads the Dirty Value(i.e modified but not commited) then you may get some inconsistent result.

2-To avoid Dirty Reads,you have to lock the Column(Cell)

3-To Lock the Column you have to apply Isolation level **READ\_COMMITED**

**2-Repetable Read Problem->**1-when a Transaction is reading the same row repeately,you may get different set of values in different reads.this kind of problem is called Repetable Read Problem.

2-To avoid Repetable Read Problem,you have to lock the row.

3- To Lock the row you have to apply Isolation level **REPEATABLE\_READ**.

.

**3-Phantom Read Problem->**1- when a Transaction is reading the set of rows repeatedly,you may get different set of rowsin different reads. this kind of problem is called Phantom Read Problem.

2-To avoid Phantom Reads,you have to lock the Entire Table.

3- To Lock the Table you have to apply Isolation level **SERIALIZABLE.**

**Q-45:How Many Type of Transaction.**

**A:**There are two type of Transaction

1. Local Transaction
2. Distributed Transaction

**Local Transaction->**when a single database is participating in the transaction Operation(i.e DB operation) then it is called Local Transaction

EX:

Transfer the funds from one account to another account where two accounts are in the same bank or same database

**Distributed Transaction->**when two or more database is participating in the transaction Operation then it is called Distributed Transaction

EX:

Transfer the funds from one account to another account where two accounts are in the differnt bank or different database

**Q-46: How to Manage Transaction with Hibernate**

**A:**There are two steps of managing transaction

**1-Specifying the Transactional Boundaries**

Transaction tx=null;

Try

{

………….

tx=session.beginTransaction(); //TX begin

OP1;

OP2;

OP3;

tx.commit(); //TX end

}

catch(Exception e)

{

If(tx!=null)

{

Tx.rollback(); //TX end

}

}

**Specifying the Isolation Levels**

Write the following propery in hibernate.cfg.xml

**<property name=”hibernate.connection.isolation”>1/2/4/8</property>**

**Q-47: What are the Connection Provider supported by Hibernate.**

**A:** ConnectionProvider is an interface.it has the following concrete implementation

1. DriverManagerConnectionProvider
2. C3P0ConnectionProvider
3. DatasourceConnectionProvider

**Q-48: What is the default Connection Provider used in Hibernate.**

**A:** DriverManagerConnectionProvider .

To use DriverManagerConnectionProvider you need to specify the following properties in Hibernate Configuration document.

<property name

="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name

="connection.url">jdbc:mysql://localhost:3306/subhag</property>

<property name="connection.username">root</property>

<property name="connection.password">root</property>

<property name="connection.pool.size">99</property>

**Q-49: What is Connection Pooling.**

**A: Connection pooling** is a pattern used by software applications to connect to databases using a pre-created set of reusable **connection** objects. When a new **connection** is required, an existing **connection** is retrieved from the **pool**.

**Q-50:How to achieve Connection Pooling in Hibernate**

**A:** When we want to achieve Connection Pooling in Hibernate we use **C3P0 Connection Provider**

1. C3P0 is Third-Party Connection pooling technique which can be used in any kind of Application
2. When you want to use C3P0 Connections , do the following
   * Add c3p0-0.9.1.jar to project build path
   * You need to specify the following properties in Hibernate configuration document.

<hibernate-configuration>

 <session-factory>

<property name="hibernate.connection.**driver\_class"**>com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.**url"**>jdbc:mysql://localhost:3306/myschema</property>

<property name="hibernate.connection.**username"**>user</property>

<property name="hibernate.connection.**password"**>password</property>

<property name="hibernate.**dialect"**>org.hibernate.dialect.MySQLDialect

</property>

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

<property name="hibernate.**c3p0.min\_size"**>5</property>

<property name="hibernate.**c3p0.max\_size"**>20</property>

<property name="hibernate.**c3p0.timeout"**>300</property>

<property name="hibernate.c3p0.**max\_statements"**>50</property>

<property name="hibernate.c3p0**.idle\_test\_period"**>3000</property>

  . . . .

</session-factory>

**hibernate.c3p0.min\_size** – Minimum number of JDBC connections in the pool. Hibernate default: 1  
**hibernate.c3p0.max\_size** – Maximum number of JDBC connections in the pool. Hibernate default: 100  
**hibernate.c3p0.timeout** – When an idle connection is removed from the pool (in second). Hibernate default: 0, never expire.  
**hibernate.c3p0.max\_statements** – Number of prepared statements will be cached. Increase performance. Hibernate default: 0 , caching is disable.  
**hibernate.c3p0.idle\_test\_period** – idle time in seconds before a connection is automatically validated. Hibernate default: 0

**Q-49: What are the Transaction supported by Hibernate.**

**A:** Transaction is an interface.it has the following concrete implementation

1. JDBCTransaction
2. JTATransaction
3. CMTTransaction

**Q-50: What is the default Transaction used in Hibernate.**

**A:** JDBCTransaction

**Q-51: How to begin the transaction**

**A:** tx=session.beginTransaction();

**Q-52: When Transaction will be ended.**

**A:** Transaction will be ended when you call tx.commit() or rollback()

**Q-53: What are the isolation level**

**A:** There are 4 type of isolation level

1-READ\_UNCOMMITED 1

2-READ\_COMMITED 2

3-REPEATABLE\_READ 4

4-SERIALIZABLE 8

**Q-54: What is hibernate caching? How many type of caching.**

**A:** 1-Cache is representation of Database near to application.

**2-**Cache is the reusable piece of information

**3-**When you Cache the Read-mostly data,you can reduce the number of Round trip b/w your application server and database server which Increase performance of your application.

**4-**Therearetwo level of Cache

First Level Cache

Second Level Cache

**Q-55:Expalin the cache scopes.**

**A:**in General,you can have 3 type of cache scopes

1. Transactional Scope cache
2. Process Scope cache
3. Clustered Scope cache

**Transactional Scope cache->**

**1-**This kind of cache will be created when ever the transaction is started and run as long as transaction is running

**2-**Transactional Scope cache will be destroyed when ever transaction ends either by commit or by rollback

**3-**Hibernate Session cache is Transactional Scope cache.

**Process Scope cache->**

**1-**This kind of cache can be accessed by multiple Transaction running in the Application or process.

**2-**All txn sharing 1 caching only for similar result.

**3-**Hibernate Query cache is Process Scope cache.

**Clustered Scope cache->**

**1-**When your enterprise application is large-scale application then you must use cluster environment where multiple servers will be clustered and integrated with LBS for routing the requests.

**2-**in the case of Clustered Scope,when one node caches the data then same data will be replicated to other nodes automatically.

**3-**Hibernate Query cache is Clustered Scope cache

**Q-56:Expalin First Level caching in Hibernate?**

**A:1-**Hibernate first level cache is associated with the Session object.

**2-**Hibernate first level cache is enabled by default and there is no way to disable it.

**3-**Any object cached in a session will not be visible to other sessions and when the session is closed, all the cached objects will also be lost.

**4-**When ever you insert or update or select or delete the records then those

Persistence objects will be placed automatically in session cache.

session.save();

session.update();

session.delete(); ->it will delete the complete record

session.load();

**5-**youcan remove the Persistence objects from session cache using

evict(obj);->specified object from session cache

clear(); ->all the object from session cache

**Q-57:Explain Second Level caching in Hibernate?**

**A:1-**Query cache is the Second Level Cache

**2-**Query cache is disabled by default.you have to enable explicitly if

Required.

**3-**When ever you execute the Hibernate Queries(HQL,QBC..-Select SQL Statement),then all the records returned by Select Statement will be Placed in Query cache.

**Q-58: How to enable Query Cache. Write the syntax also.**

**A:**

**1-**Enable the Query Cache by writing the following property in Hibernate

configuration doc.

**<property name=”hibernate.cache.use\_query\_cache”>true</property>**

**2-**Specify the Required Cache Provider in hibernate configuration doc.

**<propert name=”hibernate.cache.provider\_class”>**

**org.hibernate.cache.EhCacheProvider**

**</property>**

**Note:**Cache provider supported by hibernate

**A:**EhCacheProvider

**B:**HashtableCacheProvider

**C:**OSCacheProvider

**D:**TreeCacheProvider

**3-**Annotate entity beans with @Cache annotation and caching strategy to use

**The caching strategy value can be one of the following**

**A:** read - only

**B:** nonstrict-read-write

**C:** read-write

**D:** transactional

**Ex:**

@Entity

@Table(name = "ADDRESS")

@Cache(usage=CacheConcurrencyStrategy.READ\_ONLY, region="employee")

public class Address

{

}

4-Create EHCache configuration file, a sample file would look like below.

**ehcache.xml->(**for storage purpose**)**

<ehcache>

<diskStore path="java.io.tmpdir" />

<defaultCache

maxEntriesLocalHeap="100"

eternal="false"

timeToIdleSeconds="120"

timeToLiveSeconds="120"

overflowToDisk="true" />

<cache name="employee"

maxElementInMemory="100"

eternal="false"

timeToIdleSeconds="5"

timeToLiveSeconds="10">

overflowToDisk="true" />

</ehcache>

**Q-59:What are different ways to disable hibernate second level cache?**

**A:** 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

**Q-60: What is the default cache service of hibernate?**  
 **A:**Hibernate supports multiple cache services like **EHCache,OSCache,SWARMCache and TreeCache** and **default cache service of hibernate is EHCache.**

**Q-61: What are different states of Object in Hibernate?**

**A:** Persistence class Object can be found in 3 states.

1. Transient State
2. Persistent State
3. Detached State

**1-Transient State->**

**1-**When Persistence class Object is newly created and not participated in any Session operation then that object is called as Transient Object

**2-**State of that object is called Transient State.

**3-**Transient Object does not contain any identity or Primary Key.

**2-Persistent State->**

**1-**When Persistence class Object is participated in any Session operation then that object is called as Persistent Object

**2-**State of that object is called Transient State.

**3-**Persistent Object contain any identity or Primary Key.

**4-**Any modification happened on the Persistent Object will be reflected to DB

**3-Detached State->**

**1-**When Persistence class Object is participated in any Session operation and Remove form Session cache then that object is called as Detached Object

**2-**State of that object is called Transient State.

**3-**Persistent Object contain any identity or Primary Key.

**4-**Any modification happened on the Persistent Object will not be reflected to DataBase.

**Q-62: How can we reattach any detached objects in Hibernate?**

**A:**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.

Using:- merge(),saveUpdate()

**Q-63: How can be remove Persistence Object from Session Cache.**

**A:** There are two method to remove the Persistence Object from

Session Cache.

1-evivt(obj)

2-clear()

Close()

**Q-64: What is the diff b/w evict(obj) and clear().**

**A:**  evict(obj);->it will remove specified object from session cache

clear(); ->it will remove all the object from session cache

**Q-65:What is cascading and what are different types of cascading available?**

**A:**Cascade is a process of applying operation happened on the parent to all the childs automatically.

Cascading means that if you insert, update or delete an object, related objects are inserted, updated or deleted as well. If you do not use cascade you would have to save both objects independently. If you initially create objects and you do not cascade then you must save each object explicitly.

**Hibernate supports following cascade styles**

1-none

2-save

3-update

4-save-update

5-delete

6-all

**You can specify the required cascade style using cascade attribute as follows :**

<class name =”Student” table = “students”>

………..

<list name = ”emails” table = ”emails” **cascade = “delete”**>

………..

</list>

</class>

Here,when student record is deleted then automatically emails related to that students will be deleted.

**Q-66.Define cascade and inverse option in one-many mapping?**

**A:**cascade - enable operations to cascade to child entities.  
cascade="all|none|save-update|delete|all-delete-orphan"  
  
**inverse - mark this collection as the "inverse" end of a bidirectional association.**  
inverse="true|false"   
Essentially "inverse" indicates which end of a relationship should be ignored, so when persisting a parent who has a collection of children, should you ask the parent for its list of children, or ask the children who the parents are?

**Q-67:What is Hibernate Proxy and how it helps in lazy loading?**

**A:**

1. The Hibernate proxy is used to hide the lazy association initialization.
2. The proxy class is generated at runtime and it extends the original entity class.
3. For basic properties it simply delegates the call to the original entity.
4. Every List, Set, Map type is substituted by a PersistentList, PersistentSet, PersistentMap. These classes are responsible for intercepting a call to an uninitialized collection.

**Q-68:How to integrate Hibernate with Spring frameworks?**

**A:**

**Q-69:How can we reduce database write action times in Hibernate?**

**A:** 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.

**Q-70:What’s the usage of callback interfaces in hibernate?**

**A:** 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.

**Q-71:What’s the use of version property in hibernate?**

**A:** Version property is used in hibernate to know whether an object is in transient state or in detached state

**Q-72:What are POJOs and what’s their significance?**

**A:** 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.

**Q-73: What is the diff b/w java bean and pojo class**

**A:**

A JavaBean is a Java object that satisfies certain programming conventions:

* the JavaBean class must implement either Serializable or Externalizable;
* the JavaBean class must have a no-arg constructor;
* all JavaBean properties must have public setter and getter methods (as appropriate);
* all JavaBean instance variables should be private.

**Q-74: Which design patterns are used in Hibernate framework?**

**A:** Some of the design patterns used in Hibernate Framework are:

**Domain Model Pattern** – An object model of the domain that incorporates both behavior and data.

**Data Mapper** – A layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself.

**Proxy Pattern** – for lazy loading

**Factory pattern** – in Session Factory

**Q-75:What is HibernateTemplate class?**

**A:**Thespring framework provides HibernateTemplate (org.springframework.orm. hibernate.HibernateTemplate) which is kind of helper class and provides following benefits.

-HibernateTemplate class simplifies interaction with Hibernate session  
-Common functions are simplified to single method calls.

-Sessions are automatically closed.

-Exception are automatically caught and converted to runtime exceptions .

**Q-76: What is the default auto commit value in Hibernate**

**A:** false

**Q-77: .What are the ways to express joins in HQL?**  
 A:HQL provides four ways of expressing (inner and outer) joins:-

* An implicit association join
* An ordinary join in the FROM clause
* A fetch join in the FROM clause.
* A theta-style join in the WHERE clause.

**Q-78: write the code to insert the record in database using Hibernate**

**Q-79: write the code to delete the record in database using Hibernate**

**Q-80: write the code to select the record from database using based on id.**