**Ques. What is Hibernate Framework?**

Ans - Hibernate is a framework which provides some abstraction layer, meaning that the programmer does not have to worry about the implementations, Hibernate does the implementations for you internally like Establishing a connection with the database, writing query to perform CRUD operations etc.

It is a java framework which is used to develop persistence logic. Persistence logic means to store and process the data for long use. More precisely Hibernate is an open-source, non-invasive, light-weight java ORM(Object-relational mapping) framework to develop objects which are independent of the database software and make independent persistence logic in all JAVA, JEE.

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

**Ques. What is Java Persistence API (JPA)?**

Ans - JPA is not a tool or framework; rather, it defines a set of concepts that can be implemented by any tool or framework. While JPA's object-relational mapping (ORM) model was originally based on Hibernate, it has since evolved. Likewise, while JPA was originally intended for use with relational/SQL databases, some JPA implementations have been extended for use with NoSQL datastores.

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

**Ques. What is ORM?**

Ans - Object-relational mapping is a task–one that developers have good reason to avoid doing manually. A framework like Hibernate ORM codifies that task into a library or framework, an ORM layer. As part of the application architecture, the ORM layer is responsible for managing the conversion of objects to interact with the tables and columns in a relational database. In Java, the ORM layer converts Java classes and objects so that they can be stored and managed in a relational database.

By default, the name of the object being persisted becomes the name of the table, and fields become columns. Once the table is set up, each table row corresponds to an object in the application. Object mapping is configurable, but defaults tend to work well.

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

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

Ans -

1. Hibernate framework support Auto DDL operations. In JDBC manually we have to create table and declare the data-type for each and every column. But Hibernate can do DDL operations for you internally like creation of table,drop a table,alter a table etc.

2. Hibernate supports Auto Primary key generation. It means in JDBC we have to manually set a primary key for a table. But Hibernate can this task for you.

3. Hibernate framework is independent of Database because it supports HQL (Hibernate Query Language) which is not specific to any database, whereas JDBC is database dependent.

4. In Hibernate, Exception Handling is not mandatory, whereas In JDBC exception handling is mandatory.

Hibernate supports Cache Memory whereas JDBC does not support cache memory. so hibernate is faster as compared to jdbc.

5. Hibernate is a ORM tool means it support Object relational mapping. Whereas JDBC is not object oriented moreover we are dealing with values means primitive data. In hibernate each record is represented as a Object but in JDBC each record is nothing but a data which is nothing but primitive values.

**6. Sql Injection is not possible**

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

Ques. What are the advantages of Hibernate over JDBC?

Ans -

1.Database Independent

Hibernate’s code is database independent because you do not need to change the HQL queries (with a few exceptions) when you change databases like MySQL, Oracle, etc. Hence, it is easy to migrate to a new database. It is achieved by using a friendly dialect to communicate with the database. The database can be specified using a dialect in the Hibernate configuration XML as follows.

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

2.Reduce Repeat Code

Hibernate reduces the amount of repeating lines of code, which you can often find with JDBC. For your understanding, I have left a simple scenario below.

Like we create Hibernate SessionFactory once and use All the connections provided by it again and again

3.Lazy Loading

We can achieve lazy loading using Hibernate. Consider an example where there is a list of users in the user table. The identity proof documents uploaded by the users are stored in the identity\_proof table. The user has a ‘one to many’ relationships with the identity\_proof. In this case, the user is the parent class and identity\_proof is the child class. If you fetch the parent class, i.e the user, all the documents associated with the user will also be fetched. Imagine the size of each document. As the number of documents increases, the size of data to be processed also increases, and hence it will slow up the application.

With Hibernate, you can specify the fetch type for data as LAZY. If you do so when you fetch a user, documents will not be fetched.

4.Avoiding Try-Catch Blocks

JDBC will throw SQLexception, which is a checked exception. So you will be writing “try-catch” blocks in your code. Hibernate handles this by converting all JDBC exceptions to unchecked exceptions. Therefore, you need not waste your time implementing try-catch blocks.

5.Associations

It is easy to create an association between tables using Hibernate. Associations like one-to-one, one-to-many, many-to-one, and many-to-many can be achieved easily in Hibernate by using annotations, mapping the object entity of the required table.

6.Caching

Hibernate provides a caching mechanism, which helps reduce the number of hits, as much as possible, that your application makes to the database server. This will have a considerable effect regarding the performance of your application. There is no such caching mechanism available in JDBC. This is because Hibernate stores the object in session, which is available until the transaction is active. When a particular query is executed repeatedly, the value stored in the session is used. When a new transaction begins, the object is fetched again from the database and is stored in the session. In fact, two levels of caches are provided by Hibernate.

7.Object Mapping

In JDBC, you need to write code to map the object model’s data representation to a relational model and its corresponding schema. Hibernate itself maps Java classes to database tables using XML or by using annotations.

8. Connection pooling

Connection pooling is a mechanism in which the database connections, when created are stored in the cache by an external tool so that these connections can be reused from the cache when the application tries to connect to the same database in future. Connection pooling helps increase performance. We can achieve this connection pooling in Hibernate.

9.Hibernate supports Inheritance, Associations, Collections

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

Ques. Name some important interfaces of Hibernate framework?

Ans -

1.SessionFactory

2.Session

3.Query

4.Transaction

5.Criteria

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

Ques. What is hibernate configuration file?

Ans - hibernate.cfg.xml

this xml file contains the meta data for SessionFactory, and is read by Configuration Object in hibernate.

the meta infomation contained is mostly about-

1. DataBase Specific like (Driver Class , userl , username , password)

2. ORM related (which class has to be mapped to a table)

3. Hibernte Specific like(ddl generation , show Sql , connectionPooling , cache , dialect)

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

Ques. What is hibernate mapping file? - xml configuration (we have used annotions) ->> Student.hbm.xml

Ans - An xml file which holds the meta infotmation about ORM Mapping .

but we have used annotations in its place.

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

Ques. Name some important annotations used for Hibernate mapping?

Ans -

@Entity -> tell hibernate this Class has to be mapped to a Sql table

@Table -> used for custom name of dto class in sql, or defining our own schema fr the table

@Column -> used for custom name of columns , and gining constains to columns

@Id -> for telling hibernate this varrible will act aas a primaryKey for the table

@GeneratedValue -> for automatic genartion of primary keys (identity , sequence)

@OneToOne

@OneToMany

@ManyToMany

@JoinColumn- use to change colum name of forign key column

@Embedded - use to store object in Other class table

@Embedable - used on top of class which has to stored in other class table

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

Ques. What is Hibernate SessionFactory and how to configure it?

Ans -

1. SessionFactory is an interface. SessionFactory can be created by providing Configuration object, which will contain all DB related property details pulled from either hibernate.cfg.xml file or hibernate.properties file and it is responsible for making connection with the dataBase

2. We can create one SessionFactory implementation per database in any application. If your application is referring to multiple databases, then you need to create one SessionFactory per database.

3.The SessionFactory is a heavyweight object; it is usually created during application start up and kept for later use. The SessionFactory is a thread safe object and used by all the threads of an application.

4. SessionFactory Does the Mapping of Dto class and Sql tabl

5. Maintains 2nd level Cache in hiberna

6. Performs DDL operations

7. Its a factory for Session Object

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

Ques. What is Hibernate Session and how to get it?

Ans -

1. Session provides a physical connectivity between your application and database(User JDBC connection internally)

2. Session is resposible for CRUD opertions like(save, update , delete , read)

3. Session maintains 1st level cache in hibernate

4. Session is a lightWeight object and not thread safe.

5. Session maintains the life cycle of hibernate entity( Persistace , Transient , Detached )

6. Session is factory for Transaction , Creteria .

7. Session must be close after it is used (session.close())

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

Ques. What is difference between openSession and getCurrentSession?

Ans - Both these methods are use to obtain Session objects

sessionFactory.getCurrentSession() -> this method will return the session which is already is in use or which has ben open and not closed, if there is no session open session then this will method will create a new session object and will return it.

sessionFactory.openSession() -> this method will always create a new sessin object Object.

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

Ques. What is difference between Hibernate Session get() and load() method?

Ans -

load(UserDefined.class , new Integer(pk)) || get(UserDefined.class , new Integer(pk))

Hibernate Session class provides two method to access object e.g. session.get() and session.load() both looked quite similar to each other but there are subtle difference between load and get method which can affect performance of application.

1. Main difference between get() vs load method is that get() involves database hit if object doesn't exists in Session Cache and returns a fully initialized object which may involve several database call while load method can return proxy in place and only initialize the object or hit the database if any method other than getId() is called on persistent or entity object.

2. get() return null if no entery is available in db with given id(PK) , where s load() method throws exception(ObjectNotFoundException ).

3. Get method never returns a proxy, it either returns null or fully initialized Object, while load() method may return proxy, which is the object with ID but without initializing other properties, which is lazily initialized. If you are just using returned object for creating relationship and only need Id then load() is the way to go.

4. load() is from Jpa where as get() is hibernate specific ,

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

Ques. What is hibernate caching? Explain Hibernate first level cache?

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

there are 3 types of hibernate cache

1. 1st level (maintanied by Session)

2. 2nd level (maintanied by SessionFactory)

3. Query cache

First level cache -

First level cache in hibernate is enabled by default and you do not need to do anything to get this functionality working. In fact, you can not disable it even forcefully.

Its easy to understand the first level cache if we understand the fact that it is associated with Session object. As we know session object is created on demand from session factory and it is lost, once the session is closed. Similarly, first level cache associated with session object is available only till session object is live. It is available to session object only and is not accessible to any other session object in any other part of application.

- First level cache is associated with “session” object and other session objects in application can not see it.

- The scope of cache objects is of session. Once session is closed, cached objects are gone forever.

- First level cache is enabled by default and you can not disable it.

- When we query an entity first time, it is retrieved from database and stored in first level cache associated with hibernate session.

- If we query same object again with same session object, it will be loaded from cache and no sql query will be executed.

- The loaded entity can be removed from session using evict() method. The next loading of this entity will again make a database call if it has been removed using evict() method.

- The whole session cache can be removed using clear() method. It will remove all the entities stored in cache.

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

Ques. What are different states of an entity in hibernate?

Ans - an entity can be in three states, transient, persistent, and detached. When an object is in transient state, it is commonly refereed as transient object, similarly if it is in persistence and detached state, it is known as persistent and detached object. When an entity is first created using the new operator e.g. new User() and not associated with Hibernate session e.g. you haven't called session.save(user) method then it is known as transient object. At this stage, Hibernate doesn't know anything about this object and the object doesn't have any representation in database e.g. a corresponding row in the User table. Hibernate will not run any SQL query to reflect any changes on this object. You can move this object into persistent state by associating it with an hibernate session e.g. by calling save() or saveOrUpdate() method from an hibernate Session.

When an entity object moved to Persistence state it become responsibility of Hibernate. Now if you make any change on entity object e.g. change any attribute like user.setName("Kshitij"), Hibernate will automatically run the update queries to persist the change into database. A persistence object has corresponding representation on the database.

When you close the hibernate session or call the evict() method then the object moves to the detached state. In this state, hibernate doesn't track the object but you can re-attach a detached object to Hibernate session by calling the update() or saveOrUpdate(), or merge() method. Once reattached, the detached object will move to Persistent state.

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

Ques. What is difference between Hibernate save() and persist() methods?

Ans -

1. save() is hibernate methods where as persist() comes from Jpa

2. save() might work outside Transaction boundary sometimes depending upon how hibernate is generating the pk

where as persist() will never work outside Transaction boundary

3. Serializable save () vs void persist()

save() return the pk generated where as persist() doesnt not return anything

4. save() first generate pk then save object where as there is not doc which tells when the pk is generated in case of persist()

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

Ques. What will happen if we don’t have no-args constructor in Entity class?

Ans- Hibernate uses Reflection API to create instance of Entity beans. The method Class. ... So if we won't have no-args constructor in entity beans, hibernate will fail to instantiate it and we will get HibernateException.

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

Ques. Why we should not make Entity Class final?

Ans - As hibernate internall uses Proxy design patter to create proxy object in some cases like load() method or in lazy loading concept and proxy design patter is based on inheritance and final class is opposite of inheritance

so we dont make our final in hibernate at all , other wise hbernate will throw an exception

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

Ques. What is HQL and what are it’s benefits?

Ans - Hibernate Query Language (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. HQL queries are translated by Hibernate into conventional SQL queries, which in turns perform action on database.

Although you can use SQL statements directly with Hibernate using Native SQL, but I would recommend to use HQL whenever possible to avoid database portability hassles, and to take advantage of Hibernate's SQL generation and caching strategies.

Keywords like SELECT, FROM, and WHERE, etc., are not case sensitive, but properties like table and column names are case sensitive in HQL.

Hql is database independent language and can use for varrious takes like

-Apply restrictions to properties of objects

-Arrange the results retuned by a query by using the order by clause

-Paginate the results

-Aggregate the records by using group by and having clauses

-Use Joins

-Create user-defined functions

-Execute subqueries

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

Ques. Can we execute native sql query in hibernate?

Ans - Yes , we cane a method anmed as

SQLQuery query = session.createSqlQuery("select \* from table\_user");

List<Object[]> rows = query.list();

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

Ques. What is lazy nad eager loading in hibernate?

Ans -

Eager Loading is a design pattern in which data initialization occurs on the spot

Lazy Loading is a design pattern which is used to defer initialization of an object as long as it's possible

@OneToMany(fetch = FetchType.LAZY)

@OneToMany(fetch = FetchType.EAGER)

loading in relationships ->

OneToMany: LAZY

ManyToOne: EAGER

ManyToMany: LAZY

OneToOne: EAGER

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

Ques. How to implement relationships in hibernate?

Ans -

Using assocaition (has-a) -

(one to one , one to many , many to many)

Using inheritance (is-a) -

(Table Per Hierarchy,Table Per Concrete class,Table Per Subclass)

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

Ques. How transaction management works in Hibernate?

Ans -

A transaction simply represents a unit of work. In such case, if one step fails, the whole transaction fails (which is termed as atomicity). A transaction can be described by ACID properties (Atomicity, Consistency, Isolation and Durability).

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 methods of Transaction interface are as follows:

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.

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

Ques. What is cascading and what are different types of cascading?

Ans - Cascading can be said as , an action performed like ( save , updated , delete ) on one object/entity will reflected on the another object/enitity which is associated with the corsponding object.

There is no default cascade type in JPA. By default no operations are cascaded.

Cascade types supported by the Java Persistence Architecture are as below:

CascadeType.PERSIST : cascade type presist means that save() or persist() operations cascade to related entities.

CascadeType.MERGE : cascade type merge means that related entities are merged when the owning entity is merged.

CascadeType.REFRESH : cascade type refresh does the same thing for the refresh() operation.

CascadeType.REMOVE : cascade type remove removes all related entities association with this setting when the owning entity is deleted.

CascadeType.DETACH : cascade type detach detaches all related entities if a “manual detach” occurs.

CascadeType.ALL : cascade type all is shorthand for all of the above cascade operations.

eq

class Student{

@OneToOne(cascade = CascadeType.PERSIST)

private Laptop laptop

}

Student stundet = new Student();

Laptop laptop = new Laptop()

student.setLaptop(laptop);

so if i do

session.save(student);

then laptop will also gets saved automatically

we can pass multiple cascade type by

@OneToOne(cascade={CascadeType.REFRESH, CascadeType.MERGE})