What is Hibernate?

Hibernate is a powerful, high performance object/relational persistence and query service.

This lets the users to develop persistent classes following object-oriented principles such as association, inheritance, polymorphism, composition, and collections.

What is ORM?

ORM stands for Object/Relational mapping. It is the programmed and translucent perseverance of objects in a Java application in to the tables of a relational database using the metadata that describes the mapping between the objects and the database. It works by transforming the data from one representation to another.

What does an ORM solution comprises of?

• It should have an API for performing basic CRUD (Create, Read, Update, Delete) operations on objects of persistent classes

• Should have a language or an API for specifying queries that refer to the classes and the properties of classes

• An ability for specifying mapping metadata

• It should have a technique for ORM implementation to interact with transactional objects to perform dirty checking, lazy association fetching, and other optimization functions

What are the different levels of ORM quality?

There are four levels defined for ORM quality.

i. Pure relational

ii. Light object mapping

iii. Medium object mapping

iv. Full object mapping

What is a pure relational ORM?

The entire application, including the user interface, is designed around the relational model and SQL-based relational operations.

What is a meant by light object mapping?

The entities are represented as classes that are mapped manually to the relational tables.The code is hidden from the business logic using specific design patterns. This approach is successful for applications with a less number of entities, or applications with common, metadata-driven data models. This approach is most known to all.

What is a meant by medium object mapping?

The application is designed around an object model. The SQL code is generated at build time. And 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. Used when the mapping exceeds 25 different database products at a time.

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 are the benefits of ORM and Hibernate?

There are many benefits from these. Out of which the following are the most important one.

i. Productivity – Hibernate reduces the burden of developer by providing much of the functionality and let the developer to concentrate on business logic.

ii. Maintainability – As hibernate provides most of the functionality, the LOC for the application will be reduced and it is easy to maintain. By automated object/relational persistence it even reduces the LOC.

iii. Performance – Hand-coded persistence provided greater performance than automated one. But this is not true all the times. But in hibernate, it provides more optimization that works all the time there by increasing the performance. If it is automated persistence then it still increases the performance.

iv. Vendor independence – Irrespective of the different types of databases that are there, hibernate provides a much easier way to develop a cross platform application.

How does hibernate code looks like?

```
Session session = getSessionFactory().openSession();
Transaction tx = session.beginTransaction();
MyPersistanceClass mpc = new MyPersistanceClass ("Sample App");
session.save(mpc);
tx.commit();
session.close();
```

The Session and Transaction are the interfaces provided by hibernate. There are many other interfaces besides this.

What is a hibernate xml mapping document and how does it look like?

In order to make most of the things work in hibernate, usually the information is provided in an xml document. This document is called as xml mapping document. The document defines, among other things, how properties of the user defined persistence classes' map to the columns of the relative tables in database.

```
<?xml version="1.0"?> <!DOCTYPE hibernate-mapping PUBLIC
"http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
<hibernate-mapping>
<class name="sample.MyPersistanceClass" table="MyPersitaceTable">
<id name="id" column="MyPerId">
```

```
<generator class=”increment”/>
</id>
<property name=”text” column=”Persistance_message”/>
<many-to-one name=”nxtPer” cascade=”all” column=”NxtPerId”/>
</class>
</hibernate-mapping>
```

Everything should be included under <hibernate-mapping> tag. This is the main tag for an xml mapping document.

What the Core interfaces are of hibernate framework?

There are many benefits from these. Out of which the following are the most important one.

i. 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.

ii. 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.

iii. 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.

iv. 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.

v. 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 are Extension interfaces?

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 called as Extension interfaces.

What are the Extension interfaces that are there in hibernate?

There are many extension interfaces provided by hibernate.

ProxyFactory interface – used to create proxies
ConnectionProvider interface – used for JDBC connection management
TransactionFactory interface – Used for transaction management
Transaction interface – Used for transaction management
TransactionManagementLookup interface – Used in transaction management.
Cahce interface – provides caching techniques and strategies
CacheProvider interface – same as Cache interface
ClassPersister interface – provides ORM strategies
IdentifierGenerator interface – used for primary key generation
Dialect abstract class – provides SQL support

What are different environments to configure hibernate?

There are mainly two types of environments in which the configuration of hibernate application differs.

i. Managed environment – In this kind of environment everything from database connections, transaction boundaries, security levels and all are defined. An example of this kind of environment is environment provided by application servers such as JBoss, Weblogic and WebSphere.

ii. Non-managed environment – This kind of environment provides a basic configuration template. Tomcat is one of the best examples that provide this kind of environment.

What is the file extension you use for hibernate mapping file?

The name of the file should be like this : filename.hbm.xml.The filename varies here. The extension of these files should be “.hbm.xml”.This is just a convention and it’s not mandatory. But this is the best practice to follow this extension.

What do you create a SessionFactory?

```
Configuration cfg = new Configuration();
cfg.addResource(“myinstance/MyConfig.hbm.xml”);
cfg.setProperties( System.getProperties() );
SessionFactory sessions = cfg.buildSessionFactory();
```

First, we need to create an instance of Configuration and use that instance to refer to the location of the configuration file. After configuring this instance is used to create the SessionFactory by calling the method buildSessionFactory().

What is meant by Method chaining?

Method chaining is a programming technique that is supported by many hibernate interfaces. This is less readable when compared to actual java code. And it is not mandatory to use this format. Look how a SessionFactory is created when we use method chaining.

```
SessionFactory sessions = new Configuration()
sessions.addResource(“myinstance/MyConfig.hbm.xml”)
sessions.setProperties( System.getProperties() ).buildSessionFactory();
```

What does hibernate.properties file consist of?

This is a property file that should be placed in application class path. So when the
Configuration object is created, hibernate is first initialized. At this moment the

application will automatically detect and read this hibernate.properties file.
hibernate.connection.datasource = java:/comp/env/jdbc/AuctionDB
hibernate.transaction.factory_class =
net.sf.hibernate.transaction.JTATransactionFactory
hibernate.transaction.manager_lookup_class =
net.sf.hibernate.transaction.JBossTransactionManagerLookup
hibernate.dialect = net.sf.hibernate.dialect.PostgreSQLDialect

What should SessionFactory be placed so that it can be easily accessed?

As far as it is compared to J2EE environment, if the SessionFactory is placed in JNDI then it can be easily accessed and shared between different threads and various components that are hibernate aware. You can set the SessionFactory to a JNDI by configuring a property hibernate.session_factory_name in the hibernate.properties file.

What are POJOs?

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. Hibernate applications works efficiently with POJOs rather then simple java classes.

What is object/relational mapping metadata?

ORM tools require a metadata format for the application to specify the mapping between classes and tables, properties and columns, associations and foreign keys, Java types and SQL types. This information is called the object/relational mapping metadata. It defines the transformation between the different data type systems and relationship representations.

What is HQL?

HQL stands for Hibernate Query Language. Hibernate allows the user to express queries in its own portable SQL extension and this is called as HQL. It also allows the user to express in native SQL.

What are the different types of property and class mappings?

• *Typical and most common property mapping*
<property name="description" column="DESCRIPTION" type="string"/>
Or
<property name="description" type="string">
column name="DESCRIPTION"/>
</property>
• *Derived properties*
<property name="averageBidAmount" formula="(select AVG(b.AMOUNT) from BID b
where b.ITEM_ID = ITEM_ID)" type="big_decimal"/>
• *Typical and most common property mapping*
<property name="description" column="DESCRIPTION" type="string"/>
• *Controlling inserts and updates*
<property name="name" column="NAME" type="string"
insert="false" update="false"/>

What is Attribute Oriented Programming?

XDoclet has brought the concept of attribute-oriented programming to Java. Until JDK 1.5, the Java language had no support for annotations; now XDoclet uses the Javadoc tag format (@attribute) to specify class-, field-, or method-level metadata attributes. These attributes are used to generate hibernate mapping file automatically when the application is built. This kind of programming that works on attributes is called as Attribute Oriented Programming.

What are the different methods of identifying an object?

There are three methods by which an object can be identified.
i. Object identity –Objects are identical if they reside in the same memory location in the JVM. This can be checked by using the = = operator.
ii. Object equality – Objects are equal if they have the same value, as defined by the equals( ) method. Classes that don't explicitly override this method inherit the implementation defined by java.lang.Object, which compares object identity.
iii. Database identity – Objects stored in a relational database are identical if they represent the same row or, equivalently, share the same table and primary key value.

What are the different approaches to represent an inheritance hierarchy?

i. Table per concrete class.
ii. Table per class hierarchy.
iii. Table per subclass.

What are managed associations and hibernate associations?

Associations that are related to container management persistence are called managed associations. These are bi-directional associations. Coming to hibernate associations,these are unidirectional.

What is the Session factory interface ?

It creates new hibernate sessions by referencing immutable and thread safe objects. Application using hibernate are usually allowed and desgined to implement single instance of the class using this interface. Only single instance of a class can be used which is using this interface.

What is the session interface?
This represents hibernate session which perform the manipulation on the database entities. Some of the activities performed by session interface are as follows they are managing the persistence state, fetching persisted ones and management of the transaction demarcation.

What is the Steps involved in creating database applications with Java using Hibernate ?
Creating Database applications with Java is made simpler with Hibernate. First Plain old java object needs to be written, XML mapping file should be created which shows relationship between database and class attributes. Hibernate APIs can be used to store persistent objects.
Explain how you can configure hibernate.cfg.xml ?
Hibernate can be configured with two types of files out of which hibernate.cfg.xml is widely used and popular feature. Hibernate consults hibernate.cfg.xml file for its operating properties such as database dialect, connection string and mapping files. These files are searched on class path.
Can a single Hibernate Session object be used across multiple threads ?
No, Hibernate Session is basically single-threaded and not to be used across
multiple threads.

No, Hibernate SessionFactory is basically thread-safe, thus can be re-used
across multiple threads and multiple Transacions as well.

What is session.save() : Save does an insert and will fail if the primary key is already persistent.

What is session.saveOrUpdate() : saveOrUpdate does a select first to determine if it needs to do an insert or an update. Insert data if primary key not exist otherwise update data.
What is session.persist() : Does the same like session.save().
But session.save() return Serializable object but session.persist() return void.
What is lazy loading in hibernate and how is it done ?
Lazy fetching decides whether to load child objects while loading the Parent Object.
You need to do this setting respective hibernate mapping file of the parent class.
Lazy = true (means not to load child)
By default the lazy loading of the child objects is true. This make sure that the child objects are not loaded unless they are explicitly invoked in the application by calling getChild() method on parent.In this case hibernate issues a fresh database call to load the child when getChild() is actully called on the Parent object.But in some cases you do need to load the child objects when parent is loaded.
Just make the lazy=false and hibernate will load the child when parent is loaded from the database.

In the Employee.hbm.xml file
```
<set name="address" inverse="true" cascade="delete" lazy="false">
<key column="a_id" />
<one-to-many class="beans Address"/>
</set>
```

What is the dirty checking feature of Hibernate ?
Dirty checking feature of the Hibernate allows users or developers to avoid time consuming data base write actions. This feature makes necessary updations and changes to the fields which require a change, remaining fields are left unchanged or untouched.
In Hibernate, the most common used object is 'Session', can you explain what a session is? Is there a way by which you can share a session object between different threads?
Session is nothing but a light weight and a non-threadsafe object that represents a single unit-of-work with the database. Sessions are opened by a SessionFactory and then are closed when all work is complete. Session is the primary interface for the persistence service. A session obtains a database connection lazily (i.e. only when required). To avoid creating too many sessions ThreadLocal class can be used as shown below to get the current session no matter how many times you make call to the currentSession( ) method.
```
public class HibernateUtility {
public static final ThreadLocal local = new ThreadLocal();
public static Session currentSession() throws HibernateException {
Session session = (Session) local.get();
//open a new session if this thread has no session
if(session == null) {
session = sessionFactory.openSession();
local.set(session);
```

```
}
return session;
}
}
```

It is also vital that you close your session after your unit of work completes. You cannot share it between threads.

Explain about SessionFactory in Hibernate ? Is SessionFactory thread-safe object?

SessionFactory is Hibernate's concept of a single datastore and is threadsafe so that many threads can access it concurrently and request for sessions and immutable cache of compiled mappings for a single database. A SessionFactory is usually only built once at startup. SessionFactory should be wrapped in some kind of singleton so that it can be easily accessed in an application code.

SessionFactory sessionFactory = new Configuration( ).configure( ).buildSessionfactory( );

What are the different object states in hibernate ?

Persistent objects and collections are short lived single threaded objects, which store the persistent state. These objects synchronize their state with the database depending on your flush strategy (i.e. auto-flush where as soon as setXXX() method is called or an item is removed from a Set, List etc or define your own synchronization points with session.flush(), transaction.commit() calls). If you remove an item from a persistent collection like a Set, it will be removed from the database either immediately or when flush() or commit() is called depending on your flush strategy. They are Plain Old Java Objects (POJOs) and are currently associated with a session. As soon as the associated session is closed, persistent objects become detached objects and are free to use directly as data transfer objects in any application layers like business layer, presentation layer etc.

Detached objects and collections are instances of persistent objects that were associated with a session but currently not associated with a session. These objects can be freely used as Data Transfer Objects without having any impact on your database. Detached objects can be later on attached to another session by calling methods like session.update(), session.saveOrUpdate() etc. and become persistent objects.

Transient objects and collections are instances of persistent objects that were never associated with a session. These objects can be freely used as Data Transfer Objects without having any impact on your database. Transient objects become persistent objects when associated to a session by calling methods like session.save( ), session.persist( ) etc.

In Hibernate when does an object become detached ?

```
Session session1 = sessionFactory.openSession();
Car myCar = session1.get(Car.class, carId);      //"myCar" is a persistent object at
this stage.
session1.close();     //once the session is closed "myCar" becomes a detached object
you can now pass the "myCar" object all the way upto the presentation tier. It can be
modified without any effect to your database table.
myCar.setColor("Red");//no effect on the database
When you are ready to persist this change to the database, it can be reattached to
another session as shown below:
Session session2 = sessionFactory.openSession();
Transaction tx = session2.beginTransaction();
tx.commit();  //change is synchronized with the database.
session2.update(myCar);  //detached object "myCar" gets re-attached
session2.close()
```

Explain the pros and cons of detached objects?

The pros of detached objects are – When long transactions are required due to user think-time, it is the best practice to break the long transaction up into two or more transactions. You can use detached objects from the first transaction to carry data all the way up to the presentation layer. These detached objects get modified outside a transaction and later on re-attached to a new transaction via another session.

The cons of detached objects are – When working with detached objects is quite cumbersome, and it is better not to clutter up the session with them if possible. It is better to discard them and re-fetch them on subsequent requests. This approach is not only more portable but also more efficient because – the objects hang around in Hibernate's cache anyway.Also from pure rich domain driven design perspective, it is recommended to use DTOs (DataTransferObjects) and DOs (DomainObjects) to maintain the separation between Service and UI tiers.

Explain the mechanism by which you can distinguish between transient (i.e. newly instantiated) and detached objects in Hibernate ?

Hibernate uses the "version" property, if there is one.If not uses the identifier value. No identifier value means a new object. This does work only for Hibernate managed surrogate keys. Does not work for natural keys and assigned (i.e. not managed by Hibernate) surrogate keys.You can create your own strategy with Interceptor.isUnsaved( ). When you reattach detached objects, you need to make sure that the dependent objects are reattached as well.

Hope these hibernate interview questions have helped you to improve your knowledge in hibernate. We wish you all the best for your job interviews