Hibernate Job Interview Questions And Answers

Question # 1

What is Hibernate?

Answer:-

Hibernate is a pure Java object-relational mapping (ORM) and persistence framework that allows you to map plain old Java objects to relational database tables using (XML) configuration files.Its purpose is to relieve the developer from a significant amount of relational data persistence-related programming tasks.

Question # 2

What is ORM?

Answer:-

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.

Question # 3

What does an ORM solution comprises of?

Answer:-

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

Question # 4

What is a pure relational ORM?

Answer:-

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

Question # 5

What is a meant by light object mapping?

Answer:-

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.

Question # 6

What is a meant by medium object mapping?

Answer:-

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.

Question # 7

What is meant by full object mapping?

Answer:-

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.

Question # 8

What are the benefits of ORM and Hibernate?

Answer:-

There are many benefits from these. Out of which the following are the most important one.
Productivity : Hibernate reduces the burden of developer by providing much of the functionality and let the developer to concentrate on business logic. 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.
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.
Vendor independence : Irrespective of the different types of databases that are there, hibernate provides a much easier way to develop a cross platform application.

Question # 9

What the Core interfaces are of hibernate framework?

Answer:-

There are many benefits from these. Out of which the following are the most important one.
Session Interface : This is the primary interface used by hibernate applications. The instances of this interface are lightweight and are inexpensive to create and destroy. Hibernate sessions are not thread safe.
SessionFactory Interface : This is a factory that delivers the session objects to hibernate application. Generally there will be a single SessionFactory for the whole application and it will be shared among all the application threads.

Configuration Interface : This interface is used to configure and bootstrap hibernate. The instance of this interface is used by the application in order to specify the location of hibernate specific mapping documents.
Transaction Interface : This is an optional interface but the above three interfaces are mandatory in each and every application. This interface abstracts the code from any kind of transaction implementations such as JDBC transaction, JTA transaction.
Query and Criteria Interface : This interface allows the user to perform queries and also control the flow of the query execution.

Question # 10

What are Callback interfaces?

Answer:-

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.

Question #11

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

Answer:-

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.

Question # 12

What are POJOs?

Answer:-

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.

Question # 13

What is object/relational mapping metadata?

Answer:-

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.

Question # 14

What is HQL?

Answer:-

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.

Question # 15

What are the most common methods of Hibernate configuration?

Answer:-

The most common methods of Hibernate configuration are:
* Programmatic configuration
* XML configuration (hibernate.cfg.xml)

Question # 16

What are the important tags of hibernate.cfg.xml?

Answer:-

An Action Class is an adapter between the contents of an incoming HTTP rest and the corresponding business logic that should be executed to process this rest.

Question # 17

What are the Core interfaces are of Hibernate framework?

Answer:-

People who read this also read:
The five core interfaces are used in just about every Hibernate application. Using these interfaces, you can store and retrieve persistent objects and control transactions.
* Session interface
* SessionFactory interface
* Configuration interface
* Transaction interface
* Query and Criteria interfaces

Question # 18

What role does the Session interface play in Hibernate?

Answer:-

The Session interface is the primary interface used by Hibernate applications. It is a single-threaded, short-lived object representing a conversation between the application and the persistent store. It allows you to create query objects to retrieve persistent objects.
Session session =
sessionFactory.open
Session();
Session interface
role:
* Wraps a JDBC connection
* Factory for Transaction

* Holds a mandatory (first-level) cache of persistent objects, used when navigating the object

graph or looking up objects by identifier

Question # 19

What role does the SessionFactory interface play in Hibernate?

Answer:-

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

Question # 20

What is the general flow of Hibernate communication with RDBMS?

Answer:-

The general flow of Hibernate communication with RDBMS is :

* Load the Hibernate configuration file and create configuration object. It will automatically load all hbm mapping files
* Create session factory from configuration object
* Get one session from this session factory
* Create HQL Query
* Execute query to get list containing Java objects

Question # 21

What is Hibernate Query Language (HQL)?

Answer:-

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

Question # 22

How do you map Java Objects with Database tables?

Answer:-

* First we need to write Java domain objects (beans with setter and getter). The variables should be same as database columns.
* Write hbm.xml, where we map java class to table and database columns to Java class variables. Example :

```
<hibernate-mapping>
<class name="com.test.User"  table="user">
<property
column="USER_NAME"
length="255â€³
name="userName" not-
null="true"
type="java.lang.String"/>
<property
column="USER_PASSWORD"
length="255â€³
name="userPassword" not-
```

```
null="true"
type="java.lang.String"/>
</class>
</hibernate-mapping>
```

Question # 23

What Does Hibernate Simplify?

Answer:-

Hibernate simplifies:
* Saving and retrieving your domain objects
* Making database column and table name changes
* Centralizing pre save and post retrieve logic
* Complex joins for retrieving related items
* Schema creation from object model

Question # 24

What is the difference between load() and get()?

Answer:-

load() vs. get()
load() :-
Only use the load() method if you are sure that the object exists.
load() method will throw an exception if the unique id is not found in the database. load() just returns a proxy by default and database won't be hit until the proxy is first invoked.
get():-
If you are not sure that the object exists, then use one of the get() methods. get() method will return null if the unique id is not found in the database. get() will hit the database immediately.

Question # 25

What is the difference between and merge and update?

Answer:-

Use update() if you are sure that the session does not contain an already persistent instance with the same identifier, and merge() if you want to merge your modifications at any time without consideration of the state of the session.

Question # 26

Define cascade and inverse option in one-many mapping?

Answer:-

cascade - enable operations to cascade to child entities.
cascade="all|none|sa

ve-update|delete|all-
delete-orphan"
inverse - mark this collection as the â€œinverse"
end of a bidirectional association.
inverse="true|false"
Essentially â€œinverse" 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?

Question # 27

What does it mean to be inverse?

Answer:-

It informs hibernate to ignore that end of the relationship. If the one-to-many was marked as inverse, hibernate would create a child->parent relationship (child.getParent). If the one-to-many was marked as non-inverse then a child->parent relationship would be created.

Question # 28

What do you mean by Named - SQL query?

Answer:-

Named SQL queries are defined in the mapping
xml document and called wherever required.
Example:

```
<sql-query name = â€œempdetails">
<return alias="emp"
class="com.test.Emp
loyee"/> SELECT
emp.EMP_ID AS
{emp.empid},
emp.EMP_ADDRE
SS AS
{emp.address},
emp.EMP_NAME
AS {emp.name}
FROM Employee EMP WHERE emp.NAME LIKE :name
</sql-query>
Invoke Named Query :
List people = session.getNamedQuery("empdetails")
.setString("TomBrady", name)
.setMaxResults(50)
.list();
```

Question # 29

How do you invoke Stored Procedures?

Answer:-

```
<sql-query name="selectAllEmployees_SP" callable="true">
<return alias="emp" class="employee">
<return-property name="empid" column="EMP_ID"/>
<return-property name="name" column="EMP_NAME"/>
<return-property name="address" column="EMP_ADDRESS"/>
{ ? = call selectAllEmployees() }
</return>
```

</sql-query>

Question # 30

Explain Criteria API?

Answer:-

Criteria is a simplified API for retrieving entities by composing Criterion objects. This is a very convenient approach for functionality like â€œsearch" screens where there is a variable number of conditions to be placed upon the result set.

Example :

```
List employees = session.createCriteria(Employee.class)
.add(Restrictions.like("name", â€œa%") )
.add(Restrictions.like("address", â€œBoston"))
.addOrder(Order.asc("name") )
.list();
```

Question # 31

Define HibernateTemplate?

Answer:-

org.springframework.orm.hibernate.HibernateTemplate is a helper class which provides different methods for querying/retrieving data from the database. It also converts checked HibernateExceptions into unchecked DataAccessExceptions.

Question # 32

What are the benefits does HibernateTemplate provide?

Answer:-

The benefits of HibernateTemplate are :

* HibernateTemplate, a Spring Template class simplifies interactions with Hibernate Session.
* Common functions are simplified to single method calls.
* Sessions are automatically closed.
* Exceptions are automatically caught and converted to runtime exceptions.

Question # 33

How do you switch between relational databases without code changes?

Answer:-

Using Hibernate SQL Dialects , we can switch databases. Hibernate will generate appropriate hql queries based on the dialect defined.

Question # 34

If you want to see the Hibernate generated SQL statements on console, what should we do?

Answer:-

In Hibernate configuration file set as follows:

```
<property name="show_sql">true</property>
```

Question # 35

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

Answer:-

sorted
collection
vs. order
collection
sorted
collection :-
A sorted collection is sorting a collection by utilizing the sorting features provided by the Java collections framework. The sorting occurs in the memory of JVM which running Hibernate, after the data being read from database using java comparator.
If your collection is not large, it
will be more efficient way to
sort it. order collection :-
Order collection is sorting a collection by specifying the
order-by clause for sorting this collection when retrieval. If
your collection is very large, it will be more efficient way to
sort it .

Question # 36

How will you configure Hibernate?

Answer:-

The configuration files hibernate.cfg.xml (or hibernate.properties) and mapping files *.hbm.xml are used by the Configuration class to create (i.e. configure and bootstrap hibernate) the SessionFactory, which in turn creates the Session instances. Session instances are the primary interface for the persistence service.
" hibernate.cfg.xml (alternatively can use hibernate.properties): These two files are used to configure the hibernate sevice (connection driver class, connection URL, connection username, connection password, dialect etc). If both files are present in the classpath then hibernate.cfg.xml file overrides the settings found in the hibernate.properties file.
" Mapping files (*.hbm.xml): These files are used to map persistent objects to a relational database. It is the best practice to store each object in an individual mapping file (i.e mapping file per class) because storing large number of persistent classes into one mapping file can be difficult to manage and maintain. The naming convention is to use the same name as the persistent (POJO) class name. For example Account.class will have a mapping file named Account.hbm.xml. Alternatively hibernate annotations can be used as part of your persistent class code instead of the *.hbm.xml files.

Question # 37

What is a SessionFactory? Is it a thread-safe object?

Answer:-

SessionFactory is Hibernates 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.

Question # 38

What is a Session? Can you share a session object between different theads?

Answer:-

Session is a light weight and a non-threadsafe object (No, you cannot share it between threads) 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.

Question # 39

What are the benefits of detached objects?

Answer:-

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

Question # 40

What are the pros and cons of detached objects?

Answer:-
Pros:
" 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.
Cons
" In general, working with detached objects is quite cumbersome, and better to not 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.

Question # 41

How does Hibernate distinguish between transient (i.e. newly instantiated) and detached objects?

Answer:- " 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.
" Write your own strategy with Interceptor.isUnsaved().

Question # 42

What is the difference between the session.get() method and the session.load() method?

Answer:-

Both the session.get(..) and session.load() methods create a persistent object by loading the required object from the database. But if there was not such object in the database then the method session.load(..) throws an exception whereas session.get(&) returns null.

Question # 43

What is the difference between the session.update() method and the session.lock() method?

Answer:-

Both of these methods and saveOrUpdate() method are intended for reattaching a detached object. The session.lock() method simply reattaches the object to the session without checking or updating the database on the assumption that the database in sync with the detached object. It is the best practice to use either session.update(..) or session.saveOrUpdate(). Use session.lock() only if you are absolutely sure that the detached object is in sync with your detached object or if it does not matter because you will be overwriting all the columns that would have changed later on within the same transaction.
Note: When you reattach detached objects you need to make sure that the dependent objects are reatched as well.

Question # 44

How would you reatach detached objects to a session when the same object has already been loaded into the session?

Answer:-

You can use the session.merge() method call.

Question # 45

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

Answer:-

1. You must have a default no-argument constructor for your persistent classes and there should be getXXX() (i.e accessor/getter) and setXXX(i.e. mutator/setter) methods for all your persistable instance variables.
2. You should implement the equals() and hashCode() methods based on your business key and it is important not to use the id field in your equals() and hashCode() definition if the id field is a surrogate key (i.e. Hibernate managed identifier). This is because the Hibernate only generates and sets the field when saving the object.
3. It is recommended to implement the Serializable interface. This is potentially useful if you want to migrate around a multi-processor cluster.
4. The persistent class should not be final because if it is final then lazy loading cannot be used by creating proxy objects.
5. Use XDoclet tags for generating your *.hbm.xml files or Annotations (JDK 1.5 onwards), which are less verbose than *.hbm.xml files.

Question # 46

What does ORM consists of?

Answer:-

An ORM solution consists of the followig four pieces:

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

Question # 47

What are the ORM level?

Answer:-

The ORM levels are:

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