HIBERNATE INTERVIEW QUESTIONS

1.What is ORM ?

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

2.What does ORM consists of ?

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

3.What are the ORM levels ?

The ORM levels are:

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

4.What is Hibernate?

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.

5.Why do you need ORM tools like hibernate?

The main advantage of ORM like hibernate is that it shields developers from messy SQL. Apart from this, ORM provides following benefits:

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

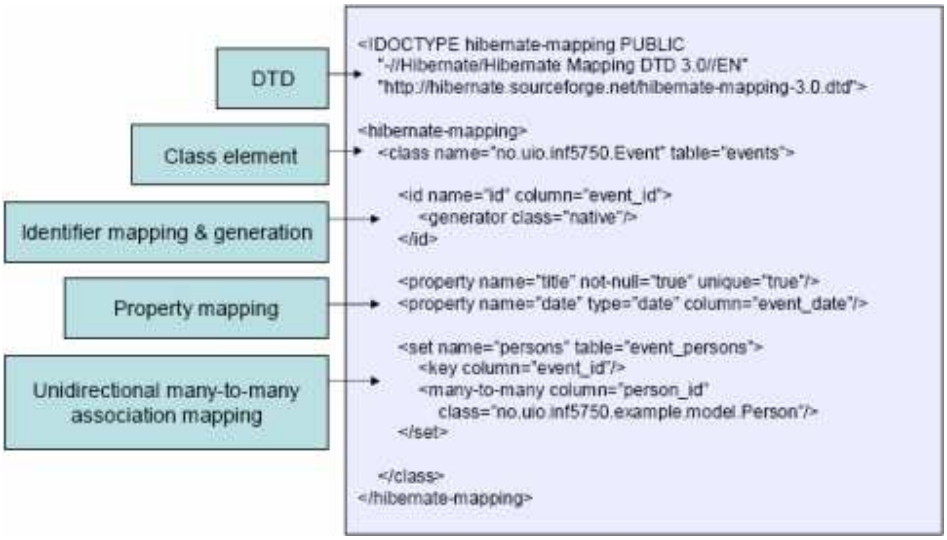
6.What Does Hibernate Simplify?

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

7.What is the need for Hibernate xml mapping file?

Hibernate mapping file tells Hibernate which tables and columns to use to load and store objects. Typical mapping file look as follows:



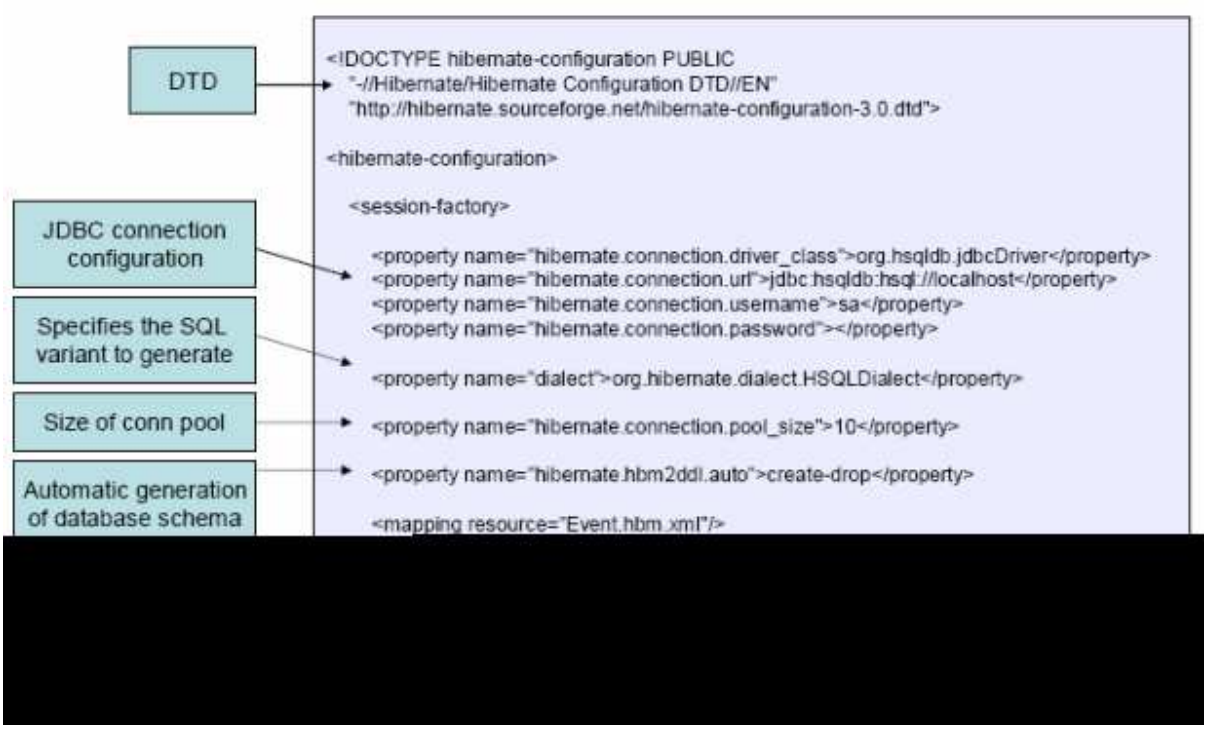
8.What are the most common methods of Hibernate configuration?

The most common methods of Hibernate configuration are:

- Programmatic configuration
- XML configuration (`hibernate.cfg.xml`)

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

Following are the important tags of hibernate.cfg.xml:



10.What are the Core interfaces are of Hibernate framework?

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

11.What role does the Session interface play in Hibernate?

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.openSession();
```

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

12.What role does the SessionFactory interface play in Hibernate?

The application obtains Session instances from a SessionFactory. There is typically a single SessionFactory for the whole applicationå¹¼reated 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();
```

13.What is the general flow of Hibernate communication with RDBMS?

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

14.What is Hibernate Query Language (HQL)?

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.

15.How do you map Java Objects with Database tables?

- First we need to write Java domain objects (beans with setter and getter).
- 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="255"
      name="userName" not-null="true"  type="java.lang.String"/>
   <property  column="USER_PASSWORD" length="255"
     name="userPassword" not-null="true"
type="java.lang.String"/>
 </class>
</hibernate-mapping>
```

16.What's the difference between load() and get()?

load() vs. get() :-

load()	get()
Only use the `load()` method if you are sure that the object exists.	If you are not sure that the object exists, then use one of the`get()` methods.
`load()` method will throw an exception if the unique id is not found in the database.	`get()` method will return null 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()` will hit the database immediately.

17.What is the difference between and merge and update ?

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

18.How do you define sequence generated primary key in hibernate?

Using <generator> tag.
Example:-

```
<id column="USER_ID" name="id" type="java.lang.Long">
   <generator class="sequence">
     <param name="table">SEQUENCE_NAME</param>
   <generator>
</id>
```

19.Define cascade and inverse option in one-many mapping?

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?

20.What do you mean by Named - SQL query?

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

```
<sql-query name = "empdetails">
   <return alias="emp" class="com.test.Employee"/>
      SELECT emp.EMP_ID AS {emp.empid},
                emp.EMP_ADDRESS 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();
```

21.How do you invoke Stored Procedures?

```
<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>
```

22.Explain Criteria API

Criteria is a simplified API for retrieving entities by composing Criterion objects. This is a very convenient approach for functionality like "search" 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", "a%") )
                     .add(Restrictions.like("address",
"Boston"))
                   .addOrder(Order.asc("name") )
                   .list();
```

23.Define HibernateTemplate?

`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.

24.What are the benefits does HibernateTemplate provide?

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.

25.How do you switch between relational databases without code changes?

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

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

In Hibernate configuration file set as follows:

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

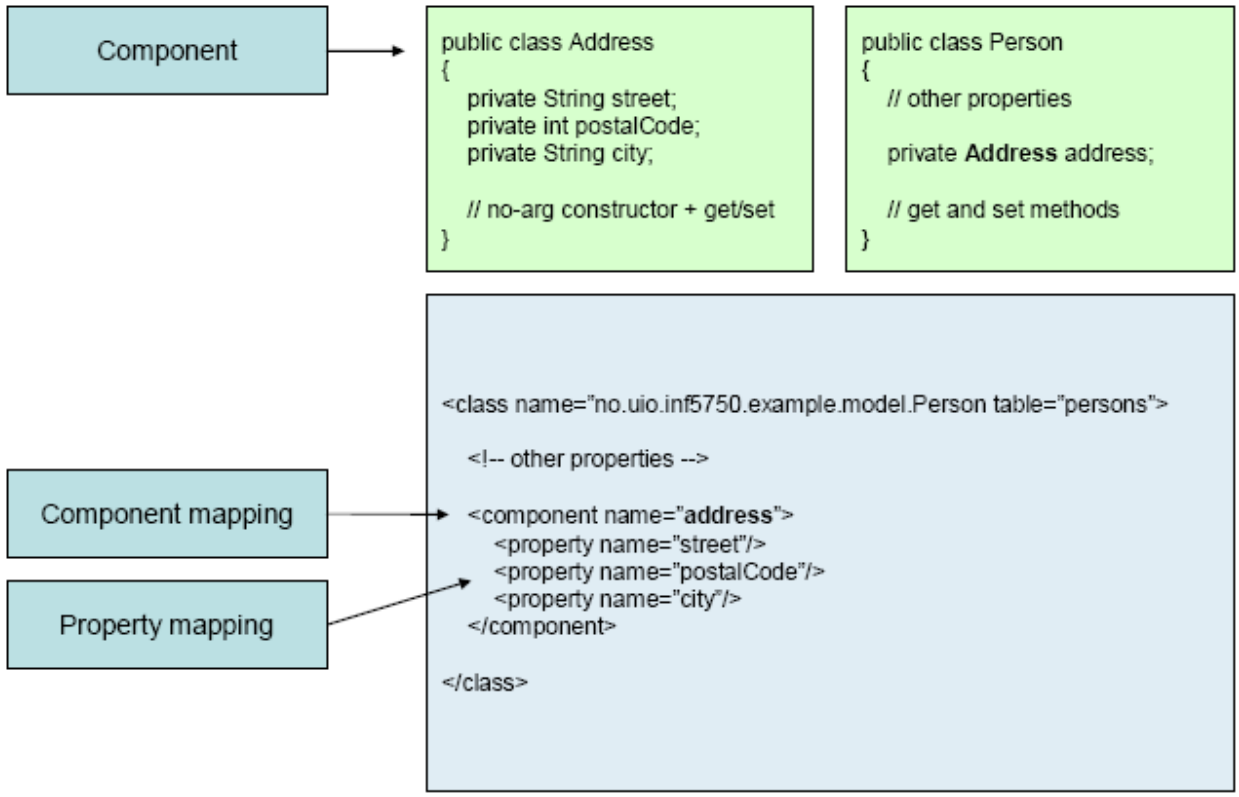
27. What are derived properties?

The properties that are not mapped to a column, but calculated at runtime by evaluation of an expression are called derived properties. The expression can be defined using the formula attribute of the element.

28. What is component mapping in Hibernate?

- A component is an object saved as a value, not as a reference
- A component can be saved directly without needing to declare interfaces or identifier properties
- Required to define an empty constructor
- Shared references not supported

Example:



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

sorted collection vs. order collection :-

sorted collection	order 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.	Order collection is sorting a collection by specifying the order-by clause for sorting this collection when retrieval.

If your collection is not large, it will be more efficient way to sort it.	If your collection is very large, it will be more efficient way to sort it .

31.What is the advantage of Hibernate over jdbc?

Hibernate Vs. JDBC :-

JDBC	Hibernate
With JDBC, developer has to write code to map an object model's data representation to a relational data model and its corresponding database schema.	Hibernate is flexible and powerful ORM solution to map Java classes to database tables. Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this.
With JDBC, the automatic mapping of Java objects with database tables and vice versa conversion is to be taken care of by the developer manually with lines of code.	Hibernate provides transparent persistence and developer does not need to write code explicitly to map database tables tuples to application objects during interaction with RDBMS.
JDBC supports only native Structured Query Language (SQL). Developer has to find out the efficient way to access database, i.e. to select effective query from a number of queries to perform same task.	Hibernate provides a powerful query language Hibernate Query Language (independent from type of database) that is expressed in a familiar SQL like syntax and includes full support for polymorphic queries. Hibernate also supports native SQL statements. It also selects an effective way to perform a database manipulation task for an application.
Application using JDBC to handle persistent data (database tables) having database specific code in large amount. The code written to map table data to application objects and vice versa is actually to map table fields to object properties. As table changed or database changed then it's essential to change object structure as well as to change code written to map table-to-object/object-to-table.	Hibernate provides this mapping itself. The actual mapping between tables and application objects is done in XML files. If there is change in Database or in any table then the only need to change XML file properties.
With JDBC, it is developer's responsibility to handle JDBC result set and convert it to Java objects through code to use this persistent data in application. So with JDBC, mapping between Java objects and database tables is done manually.	Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost.
With JDBC, caching is maintained by hand-coding.	Hibernate, with Transparent Persistence, cache is set to application work space. Relational tuples are moved to this cache as a result of query. It improves performance if client application reads same data many times for same write. Automatic Transparent Persistence allows the developer to concentrate more on business logic rather than this application code.
In JDBC there is no check that always every user has updated data. This check has to be added by the developer.	Hibernate enables developer to define version type field to application, due to this defined field Hibernate updates version field of database table every time relational tuple is updated in form of Java class object to that table. So if two users retrieve same tuple and then modify it and one user save this modified tuple to database, version is automatically updated for this tuple by Hibernate. When other user tries to save updated tuple to database then it does not allow saving it because this user does not have updated data.

32.What are the Collection types in Hibernate ?

- Bag
- Set
- List
- Array
- Map

33.What are the ways to express joins in HQL?

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.

34.Define cascade and inverse option in one-many mapping?

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?

35.What is Hibernate proxy?

The `proxy` attribute enables lazy initialization of persistent instances of the class. Hibernate will initially return CGLIB proxies which implement the named interface. The actual persistent object will be loaded when a method of the proxy is invoked.

36.How can Hibernate be configured to access an instance variable directly and not through a setter method ?

By mapping the property with access="field" in Hibernate metadata. This forces hibernate to bypass the setter method and access the instance variable directly while initializing a newly loaded object.

37.How can a whole class be mapped as immutable?

Mark the class as mutable="false" (Default is true),. This specifies that instances of the class are (not) mutable. Immutable classes, may not be updated or deleted by the application.

38.What is the use of dynamic-insert and dynamic-update attributes in a class mapping?

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

- `dynamic-update` (defaults to `false`): Specifies that `UPDATE` SQL should be generated at runtime and contain only those columns whose values have changed
- `dynamic-insert` (defaults to `false`): Specifies that `INSERT` SQL should be generated at runtime and contain only the columns whose values are not null.

39.What do you mean by fetching strategy ?

A *fetching strategy* is the strategy Hibernate will use for retrieving associated objects if the application needs to navigate the association. Fetch strategies may be declared in the O/R mapping metadata, or over-ridden by a particular HQL or `Criteria` query.

40.What is automatic dirty checking?

Automatic dirty checking is a feature that saves us the effort of explicitly asking Hibernate to update the database when we modify the state of an object inside a transaction.

41.What is transactional write-behind?

Hibernate uses a sophisticated algorithm to determine an efficient ordering that avoids database foreign key constraint violations but is still sufficiently predictable to the user. This feature is called transactional write-behind.

42.What are Callback interfaces?

Callback interfaces allow the application to receive a notification when something interesting happens to an object—for example, when an object is loaded, saved, or deleted. Hibernate applications don't need to implement these callbacks, but they're useful for implementing certain kinds of generic functionality.

43. What are the types of Hibernate instance states ?

Three types of instance states:

- Transient -The instance is not associated with any persistence context
- Persistent -The instance is associated with a persistence context
- Detached -The instance was associated with a persistence context which has been closed – currently not associated

44. What are the differences between EJB 3.0 & Hibernate

Hibernate Vs EJB 3.0 :-

Hibernate	EJB 3.0
Session-Cache or collection of loaded objects relating to a single unit of work	**Persistence Context**-Set of entities that can be managed by a given EntityManager is defined by a persistence unit
XDoclet Annotations used to support Attribute Oriented Programming	**Java 5.0 Annotations** used to support Attribute Oriented Programming
Defines HQL for expressing queries to the database	**Defines EJB QL** for expressing queries
Supports Entity Relationships through mapping files and annotations in JavaDoc	**Support Entity Relationships** through Java 5.0 annotations
Provides a Persistence Manager API exposed via the Session, Query, Criteria, and Transaction API	**Provides and Entity Manager Interface** for managing CRUD operations for an Entity
Provides callback support through lifecycle, interceptor, and validatable interfaces	**Provides callback support** through Entity Listener and Callback methods
Entity Relationships are unidirectional. Bidirectional relationships are implemented by two unidirectional relationships	**Entity Relationships are bidirectional or unidirectional**

45. What are the types of inheritance models in Hibernate?

There are three types of inheritance models in Hibernate:

- Table per class hierarchy
- Table per subclass
- Table per concrete class