**Hibernate Interview Question’s for 4-6 years**

1. **Why is ORM preferred over JDBC?**

* It allows business code access the objects rather than Database tables.
* It hides the details of SQL queries from OO logic.
* It is based on JDBC “under hood”.
* Dealing with database implementation is not required.
* Entities are based on business concepts rather than database structures.
* It generates the automatic key and Transaction management.
* Application development is faster.

1. **Mention the Key components of Hibernate?**

The Key components of Hibernate are:

* Session: It is used to get a physical network with a database.
* Transaction: It represents the unit of work with a database.
* Query: It uses SQL and HQL string to retrieve the data from the database and create objects.
* Criteria: It is used create and execute object-oriented queries and retrieve the objects.
* Configuration: It represents the properties of files required by Hibernate
* Session Factory: It configures hibernate for the application using the provided configuration file and instantiates the session object.

1. **What are the configurations involved in Hibernate framework?**

Hibernate framework is an enormous framework designed to handle almost every database operation for you. To configure such a framework involves multiple entities to be configured. These include:

* Database credentials
* Database dialect
* Database URL
* Caching levels
* ORM mapping
* Connection pool configuration
* Table creation mode – Create/Update

Hibernate comes with default values for almost every non-database configuration like the Connection pool size, caching levels, table creation mode and others. Thus, despite having so many configurable aspects, it allows to get started with minimal configuration.

1. **How do we define the primary key value generation logic?**

The primary key values can be generated using various methods depending on the database. For instance, in MySQL database the primary keys can be generated using an auto-incrementing algorithm while in Oracle database, you need a sequence to be created and used for an auto-incrementing the value for primary key. These methods of generation can be specified using the below annotation code.

@Entity

@Table(name="users")

public class User{

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

int userid;

@Column(name="user\_name")

String username;

String password;

}

The userid column here has been defined as a primary key autogenerated using the identity strategy. The possible values for strategy include:

* GenerationType.AUTO
* GenerationType.IDENTITY
* GenerationType.SEQUENCE
* GenerationType.TABLE

1. **Explain the Transaction object in Hibernate?**

It represents a unit of work with the database and most of the RDBMS (Relational Database Management System) supports transaction functionality.  
In Hibernate, transactions are managed by an underlying transaction manager and transaction from JDBC or JTA.  
It is an optional object and the Hibernate Application do not use this interface, instead, they handle the transactions in their code.

1. **What is a One-to-One association in Hibernate?**

It is similar to the many-to-one association and the difference lies in the column that will be set as a unique one.The many-to-one element is used to define one-to-one association.  
To the defined variable a name attribute is set in the parent class and the column attribute is used to set column name in the parent table, which is unique so that only one object gets associated with another.

1. **What is Many-to-Many association in Hibernate?**

The Many-to-Many element indicates the relation between one object to many other objects and column attribute is used to link intermediate columns. A Many-to-Many mapping is implemented using a Set Java collection that does not have any redundant element.

1. **What is Hibernate caching?**

Hibernate caches Query data and makes the application run faster.  
If used correctly, the hibernate cache can be very useful in achieving the faster application running performance.  
The main idea lying behind the cache is reducing the number of database queries, which results in reduced throughput time of the application.

1. **What is Query level cache in Hibernate?**

In hibernate, a cache query can be implemented that results in sets and integrates closely with the second level cache.It is an optional feature and it requires two additional cache regions that can hold the cached query results and also the timestamps whenever a table is updated. This is useful only for the queries that run frequently holding the same parameters.

1. **What are concurrency strategies?**

The concurrency strategies are the mediators who are responsible for storing items and retrieving them from the cache.In case of enabling a second level cache, the developer must decide for each persistent class and collection, and also which cache concurrency, has to be implemented.

Following are the concurrency strategies that can be implemented by the developer:

* **Transactional:** This strategy is used mostly to read data where the prevention of stale data is critical in concurrent transactions, in the unique case of an update.
* **Read- Only:** This strategy is compatible with the data that can’t be modified. We can use it for reference data only.
* **Read-Write:** It is similar to transactional strategy. where we read mostly data and prevention of stale data is critical.
* **Non-strict-Read-Write:** This strategy assures no guarantee of consistency between the database and cache. We can use this strategy only if the data can be modified and a small likelihood of stale data is not the critical concern.

1. **Describe the important interfaces of Hibernate framework?**

**Important interfaces of Hibernate framework are:**

* **SessionFactory (org.hibernate.SessionFactory)**  
  It is an immutable thread safe cache of compiled mappings for a single database.  
  We are supposed to initialize SessionFactory once and then we are allowed to cache and reuse it.  
  The SessionFactory instance is used to return the session objects for database operations.
* **Session (org.hibernate.Session)**  
  It is a single threaded and short-lived object, which represents a conversation between the persistent store and the application.  
  The session should be opened only when it is required and should be closed as soon as the user is done.  
  The session object is the interface between hibernate framework and Java application code and it provides methods for the CRUD operations.
* **Transaction (org.hibernate.transaction)**  
  It is a single threaded and short-lived object used by the application, which specifies atomic units of work.  
  The application is abstracted from the underlying JDBC or JTA transaction.

1. **What inheritance mapping strategies are available in Hibernate?**

Hibernate have 3 ways of inheritance mapping, They are

* Table per hierarchy
* Table per concrete class
* Table per subclass

1. **How to configure the Hibernate second level caching?**

The hibernate second level caching can be configured using EHCache library. To add the EHCache dependency using maven, use the following dependency code.

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-ehcache</artifactId>

<version>4.3.5.Final</version>

</dependency>

This library once added, hibernate framework needs to be configured to use the library for caching. In order to configure the second level caching using EHCache, use the below code.

<property name="hibernate.cache.region.factory\_class">org.hibernate.cache.ehcache.EhCacheRegionFactory</property>

<property name="hibernate.cache.use\_second\_level\_cache">true</property>

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

<property name="net.sf.ehcache.configurationResourceName">/myehcache.xml</property>

1. **What are different states of an entity bean?**

An element bean occurrence can exist is one of the three states.

**Transient**: When a object is never continued or related with any session, it’s in transient state. Transient cases might be influenced diligent by calling save(), persist() or saveOrUpdate(). Steady examples might be influenced transient by calling to delete().

**Persistent**: When a question is related with a one of a kind session, it’s in persevering state. Any occurrence returned by a get() or load() technique is persevering.

**Detached**: When a object is already relentless yet not related with any session, it’s in disengaged state. Disengaged occasions might be made determined by calling refresh(), saveOrUpdate(), lock() or replicate(). The condition of a transient or disengaged example may likewise be influenced relentless as another constant case by calling to merge().

1. **Is it possible to connect multiple database in a single Java application using Hibernate?**

Yes. Practically it is possible to connect a single Java application to multiple databases using two separate hibernate configuration files and two separate session factories. These hibernate configuration files contain different dialect configuration pertaining to the respective database. The entities are exclusively mapped to the relevant database configuration. Thus, with two different parallel SessionFactory objects, it is possible to have multiple databases connected.

1. **How many Hibernate sessions exist at any point of time in an application?**

Hibernate session is a shared object. During any point of time, there exists only single shared session object that helps in managing transactions and getting connections from the connection pool. It must be noted that this is true only when a single database configuration is used. In case of multiple database configurations, Hibernate would create a separate session object to maintain the mapping and transactions for the other database.

1. **Difference between get() vs load() method in Hibernate?**   
   This is one of the most frequently asked Hibernate interview questions, I have seen it several times. The key difference between the get() and load() method is that load() will throw an exception if an object with id passed to them is not found, but get() will return null.  
     
   Another important difference is that load can return proxy without hitting the database unless required (when you access any attribute other than id) but get() always go to the database, so sometimes using load() can be faster than the get() method.  
     
   Use the load() method, if you know the object exists, and get() method if you are not sure about the object's existence.
2. **What is N+1 SELECT problem in Hibernate?**   
   The N+1 SELECT problem is a result of lazy loading and load on demand fetching strategy. In this case, Hibernate ends up executing N+1 SQL queries to populate a collection of N elements.  
     
   For example, if you have a List of N Items where each Item has a dependency on a collection of Bid object. Now if you want to find the highest bid for each item then Hibernate will fire 1 query to load all items and N subsequent queries to load Bid for each item.
3. **What are some strategies to solve the N+1 SELECT problem in Hibernate?**

This is the follow-up question of previous Hibernate interview question. If you answer the last query correctly then you would be most likely asked this one.  
  
Here are some strategies to solve the N+1 problem:  
1) pre-fetching in batches, this will reduce the N+1 problem to N/K + 1 problem where  K is the size of the batch  
2) subselect fetching strategy

3) disabling lazy loading

1. **What is difference between getCurrentSession() and openSession() in Hibernate?**

An interesting Hibernate interview question as you might have used both getCurrentSession() and openSession() to obtain an instance of the Session object.

1. **When do you use merge() and update() in Hibernate?**

This is one of the tricky Hibernate interview questions. You should use update() if you are sure that the Hibernate session does not contain an already persistent instance with the same id and use merge() if you want to merge your modifications at any time without considering the state of the session.