Hibernate

Hibernate means: Sleep mode

Hibernate is for enterprise apps.

To run Hibernate JDK. JRE and Hibernate libraries are enough. So hibernate can be used for small scale and enterprise level applications application.

1. **What is Hibernate?**

Hibernate is an open-source and lightweight ORM tool that is used to store, manipulate and retrieve data from the database.

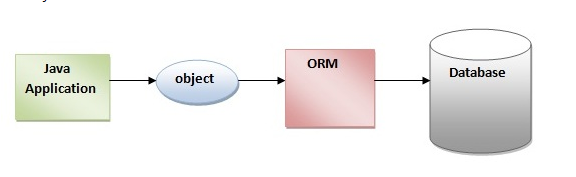
Hibernate provides framework, for mapping application domain objects to the relational database tables and vice versa.

1. **What is ORM**?

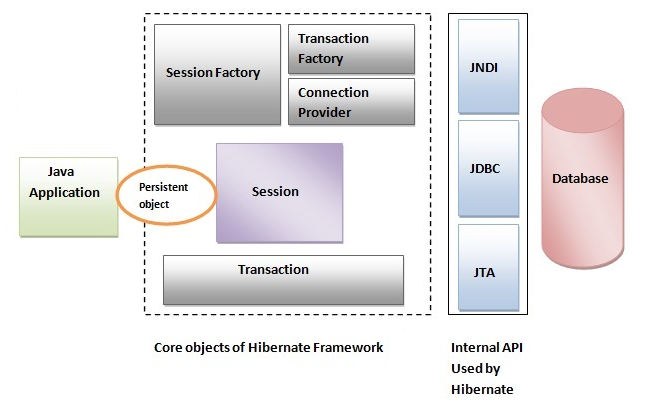
ORM (**O**bject **R**elation **M**apping) it is a programming strategy.

Using this programming strategy we can map application domain object with database table and vice versa.

Use of ORM is, it simplifies data creation, data manipulation and data access.



1. **What are the important features of Hibernate? / Advantage of Hibernate over JDBC (Remember these points base on each points can tell advantages) IMP**
2. **Cache Support**.
   1. As a result of this we get good performance.
3. **HQL (Hibernate query language)**
   1. HQL is almost similar to SQL. Database independent query
   2. HQL is fully object-oriented and understands concepts like inheritance, polymorphism and association
4. **Auto DDL** (Data Definition Language)
   1. Hibernate has capability to create the table on its own say in oracle if we pass connection properties of oracle and some configurations. It can create primary key and foreign key relationship also.
5. **Primary generator Support.**
   1. Suppose in a form they ask name age sex and other data but they do not ask ID because ID is automatically generated by the system my incrementing the previous value. In this case id will be generated by hibernate. ID is nothing but primary key of the table
6. **Transaction management:** 
   1. In case of JDBC we need to write the transactions management code separately but Hibernate supports on its own
7. **Exception handling**
   1. No need to handle any exception that is in compilation time it does not throw exception like JDBC which throws SQLException etc
8. **Data validation support.**
   1. If some form input data is not validated by controller or any where then Hibernated provides validation support. Using annotations
9. **ORM** (Object Relation Mapping) structures
   1. Is- a Relationship and has-a relationship it supports. It means we can created Objects then from objects we can define tables
   2. ORM is for RDBMS like oracle
10. **OGM** (Object Graph Mapping) support 🡪 After Hibernate4
    1. OGM is for non RDBMS like MangoDB
11. **Hibernate search support** 🡪 After Hibernate4
    1. Index based searching
12. **Connection pools built in**
13. **Hibernate Architecture?**

****

Elements of Hibernate Architecture

For creating the first hibernate application, we must know the elements of Hibernate architecture. They are as follows:

**SessionFactory**

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

**Session**

The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

**Transaction**

The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

**ConnectionProvider**

It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

**TransactionFactory**

It is a factory of Transaction. It is optional.

1. **What are the core interfaces of Hibernate?**

The core interfaces of Hibernate framework are:

SessionFactory

Session

Transaction

Configuration

Criteria

Query

1. **What is SessionFactory?**

It is a factory of Session.

It holds the data of second level cache that is not enabled by default.

1. **Is SessionFactory a thread-safe object?**

Yes, SessionFactory is a thread-safe object, many threads cannot access it simultaneously.

1. **What is Session?**

It maintains a connection between hibernate application and database

It provides methods such as persist (), update(), delete(), load(), get() etc.

It is a factory of Query, Criteria and Transaction i.e. it provides factory methods to return these instances.

1. **Is Session a thread-safe object?**

No, Session is not a thread-safe object; many threads can access it simultaneously

1. **What is the difference between session.save() and session.persist() method?**

**Save ()**

public Serializable save(Object o)

Returns the identifier (Serializable) of the instance.

**persist ()**

public void persist(Object o)

return nothing because its return type is void.

1. **What is the difference between get and load method? IMP**

**Get();**

Always hit the database.

As it hits the database it returns real object not proxy.

Returns null if object is not found.

It should be used if you are not sure about the existence of instance.

**Load()**

Doesn't hit the database.

As it does not hit the database it returns proxy object.

Throws ObjectNotFoundException if object is not found.

It should be used if you are sure that instance exists.

1. **What is the difference between update and merge method?**

**Update()**

Update means to edit something.

update() should be used if session doesn't contain an already persistent state with same id. It means update should be used inside the session only. After closing the session it will throw error.

**Merge()**

Merge means to combine something.

merge() should be used if you don't know the state of the session, means you want to make modification at any time.

Let's try to understand the difference by the example given below:

...

SessionFactory factory = cfg.buildSessionFactory();

Session session1 = factory.openSession();

Employee e1 = (Employee) session1.get(Employee.class, Integer.valueOf(101));//passing id of employee

session1.close();

e1.setSalary(70000);

Session session2 = factory.openSession();

Employee e2 = (Employee) session1.get(Employee.class, Integer.valueOf(101));//passing same id

Transaction tx=session2.beginTransaction();

session2.merge(e1);

tx.commit();

session2.close();

After closing session1, e1 is in detached state. It will not be in session1 cache. So if you call update() method, it will throw an error.

Then, we opened another session and loaded the same Employee instance. If we call merge in session2, changes of e1 will be merged in e2.

1. **What are the states of object are in hibernate? IMP**

There are 3 states of object (instance) in hibernate.

**Transient:** Object is just created but has no primary key (identifier) and not associated with session.

**Persistent:** The object is in persistent state if session is open, and you just saved the instance in the database or retrieved the instance from the database.

**Detached:** The object is in detached state if session is closed. After detached state, object comes to persistent state if you call lock () or update() method.

1. **What are the inheritance mapping strategies?**

There are 3 ways of inheritance mapping in hibernate.

Table per hierarchy

Table per concrete class

Table per subclass

1. **How to make a immutable class in hibernate?**

If you mark a class as mutable="false", class will be treated as an immutable class. By default, it is mutable="true".

1. **What is automatic dirty checking in hibernate?**

Calls update statement automatically on the objects that are modified in a transaction.

...

SessionFactory factory = cfg.buildSessionFactory();

Session session1 = factory.openSession();

Transaction tx=session2.beginTransaction();

Employee e1 = (Employee) session1.get(Employee.class, Integer.valueOf(101));

e1.setSalary(70000);

tx.commit();

session1.close();

Here, after getting employee instance e1 and we are changing the state of e1.

After changing the state, we are committing the transaction. In such case, state will be updated automatically. This is known as dirty checking in hibernate.

1. **How many types of association mapping are possible in hibernate?**

There can be 4 types of association mapping in hibernate.

One to One

One to Many

Many to One

Many to Many

1. **What is lazy loading in hibernate?**

Lazy loading in hibernate improves the performance. It loads the child objects on demand.

Since Hibernate 3, lazy loading is enabled by default, you don't need to do lazy="true". It means not to load the child objects when parent is loaded.

1. **What is HQL (Hibernate Query Language)?**

Hibernate Query Language is known as an object oriented query language.

Database independent

Simple to write query

You don't need to learn SQL

1. **What is caching in hibernate? IMP**

Caching is one of the important feature of hibernate using caching we can get better performance for our application. Caching avoids number of database hits as a result obviously our application gets better performance.

There are 2 types of caching:

**First Level Caching (Session)**

1. The first-level cache is the **Session cache** and is a **mandatory** cache; all requests must pass through this cache.

If you perform multiple updates to an object Hibernate delays the update to database as long as possible to reduce the number of update statement hit to the database.

If you close the session, all the objects being cached are lost and either persisted or updated in the database

**Second-level cache (Session factory)**

**SessionFactory** object holds the second level cache data and it is **optional.** First-level cache will always be consulted before any attempt is made to locate an object in the second-level cache.

The data stored in the second level cache will be available to entire application.

The second-level cache can be configured on a per-class and per-collection basis and mainly responsible for caching objects across sessions.

The second-level cache can be configured on a per-class and per-collection basis and mainly responsible for caching objects across sessions.

Not all classes benefit from caching, so it's important to be able to disable the second-level cache.

**Query-level cache (query resultsets)**

This cache is for **query result sets** this integrates closely with second level cache, this is **optional**

This cache requires two additional physical cache regions that hold the cached query results and the timestamps when a table was last updated.

1. **How to set up hibernate second level cache? IMP**
   1. Decide which **concurrency strategy** to use.
   2. Need to configure **cache expiration** and **physical cache attributes**  using cache provider.
2. **What is concurrency strategy in second level class? IMP**

Concurrency strategy is mediator which is responsible for storing the items of data in the cache and retrieving them from the cache.

If you are going to enable a second-level cache, you will have to decide, for each persistent class and collection, which cache concurrency strategy to use.

**read-only**: caching will work for read only operation.

**nonstrict-read-write**: caching will work for read and write but one at a time.

**read-write**: caching will work for read and write, can be used simultaneously.

**transactional**: caching will work for transaction.

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

<meta attribute="class-description">

This class contains the employee detail.

</meta>

<cache usage="read-write"/>

<id name="id" type="int" column="id">

<generator class="native"/>

</id>

<property name="firstName" column="first\_name" type="string"/>

<property name="lastName" column="last\_name" type="string"/>

<property name="salary" column="salary" type="int"/>

</class>

</hibernate-mapping>

If we are going to use second-level caching for our Employee class, let us add the mapping element required to tell Hibernate to cache Employee instances using read-write strategy.

1. **Explain about second level cache provider? List the vendors who have provided implementation for second level cache? IMP**

We can choose only one cache provider for entire application.

Below are the vendors who have provided implementation for second level cache

**EH Cache**: It can cache in memory or on disk and clustered caching and it supports the optional Hibernate query result cache.

**OS Cache**: Supports caching to memory and disk in a single JVM, with a rich set of expiration policies and query cache support.

**Swarm Cache:** A cluster cache based on JGroups. It uses clustered invalidation but doesn't support the Hibernate query cache.

**JBoss Cache**: A fully transactional replicated clustered cache also based on the JGroups multicast library. It supports replication or invalidation, synchronous or asynchronous communication, and optimistic and pessimistic locking. The Hibernate query cache is supported

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Strategy/Provider** | **Read-only** | **Nonstrictread-write** | **Read-write** | **Transactional** |
| EHCache | X | X | X |  |
| OSCache | X | X | X |  |
| SwarmCache | X | X |  |  |
| JBoss Cache | X |  |  | X |

You will specify a cache provider in hibernate.cfg.xml configuration file. We choose EHCache as our second-level cache provider:

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">

org.hibernate.dialect.MySQLDialect

</property>

<property name="hibernate.connection.driver\_class">

com.mysql.jdbc.Driver

</property>

<!-- Assume students is the database name -->

<property name="hibernate.connection.url">

jdbc:mysql://localhost/test

</property>

<property name="hibernate.connection.username">

root

</property>

<property name="hibernate.connection.password">

root123

</property>

<property name="hibernate.cache.provider\_class">

org.hibernate.cache.EhCacheProvider

</property>

<!-- List of XML mapping files -->

<mapping resource="Employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

1. **The Query-level Cache?**

To use the query cache, you must first activate it using the hibernate.cache.use\_query\_cache="true" property in the configuration file. By setting this property to true, you make Hibernate create the necessary caches in memory to hold the query and identifier sets.

Next, to use the query cache, you use the setCacheable(Boolean) method of the Query class. For example:

Session session = SessionFactory.openSession();

Query query = session.createQuery("FROM EMPLOYEE");

query.setCacheable(true);

List users = query.list();

SessionFactory.closeSession();

Hibernate also supports very fine-grained cache support through the concept of a cache region. A cache region is part of the cache that's given a name.

Session session = SessionFactory.openSession();

Query query = session.createQuery("FROM EMPLOYEE");

query.setCacheable(true);

query.setCacheRegion("employee");

List users = query.list();

SessionFactory.closeSession();

This code uses the method to tell Hibernate to store and look for the query in the employee area of the cache.

**Transactions:**

1. **How transaction management works in Hibernate? IMP**

So after getting the session from SessionFactory, we can call session.beginTransaction() to start the transaction. This method returns the Transaction reference that we can use later on to either commit or rollback the transaction.

Overall hibernate transaction management is better than JDBC transaction management because we don’t need to rely on exceptions for rollback.

Any exception thrown by session methods automatically rollback the transaction.

1. **How transaction is designed in Hibernate? What are its methods?**

**Transaction interface** that defines the unit of work. It maintains abstraction from the transaction implementation (JTA, JDBC).

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.

**void setTimeout(int seconds)** it sets a transaction timeout for any transaction started by a subsequent call to begin on this instance.

**boolean isAlive()** checks if the transaction is still alive.

**void registerSynchronization(Synchronization s)** registers a user synchronization callback for this transaction.

**boolean wasCommited()** checks if the transaction is commited successfully.

**boolean wasRolledBack()** checks if the transaction is rolledback successfully.

1. **Hibernate has @transactional annotation there will be some exception also for few of the exceptions i** do **not want to rollback how do I do that.? IMP**

By using the attribute noRollbackForClassName = "java.lang.Exception"

1. **How do you get connection in hibernate? IMP**
2. **How DMP console and console API getting connection? IMP**
3. **What method should be invoked on connection object to ensure that the user managed transaction has to be commited properly? IMP**

* References   
  Durga video

**https://www.youtube.com/watch?v=vfJe1W\_eRXk&index=2&list=PLd3UqWTnYXOlbNZyVIwgX627xvB0EWnqJ**

Demo on Hibernate - (Part - 1) by Naveen 🡪 this has history of hibernate and some comparition

First he explans what JDBC does and problems of it and solution is Ejb session then what is the problem in EJB session so some developer of EJB team found problems in EJB s to provide solution he developed Hibernate.

For running EJB you need Application server also there will be some other maintanace cost.

Jdbc is already there then Why ORM?? What is there in ORM and what is nt there in JDBC??

* If it is JDBC to communicate with oracle then in DAO class statement or prepared statement or callable statements. Then we should write queries. So those queries need to be executed in Oracle. In future if we change DB to my SQL. Then we cannot run all queries in MySQL. SO we cannot migrate the DB so our code is database dependent.
* JDBC does not support caching but hibernate supports caching.
* 🡪 JDBC table relation mapping hibernate object relartion mapping

Demo on Hibernate - (Part - 2) by Naveen -- >

Developer provided features of EJB into Hibernate and provided many other features.

Expalins Hibernate features mainly

Hibernate Object States part 1 🡪

Object types:

There are three types of objects state. Transient object , Persistant object, detached object and

Before attaching the object to hibernate that state is called as transient state because object can change at any time. Once the object is sent to hibernate then object moves to persistent scope from hibernate if we remove then object id in detached state.

Hibernate has

Basic hibernate application:

Files required are:

Pojo class,

Mapping file,

Configuration file,

Test class

How to write pojo class: pojo class need to contain required number of properties which are stored in to database table. SO it is like whatever columns we need to store should be in object

Every property must have getter and setter method.access of the setter and getter should be public.

Can have extra variables and methods and constructor as per our requirement.

Why mapping class.

When we pass object to hibernate, it needs to store the data in table. So there sould be mapping file which maps attributes and column names.